On the Efficacy of Knowledge Graph Completion Methods, Accuracy Measures and Evaluation Protocols

by

Farahnaz Akrami

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2021

To my parents with love and deepest gratitude

ACKNOWLEDGEMENTS

There are numerous people whose support, ideas, and encouragement contributed to this dissertation. Words cannot do justice to how grateful I am for all their generous support and guidance.

I would like to express my sincere gratitude to my supervisor, Dr. Chengkai Li, for providing a thriving research environment where this work could advance. His invaluable advice and unwavering support have been instrumental in conducting this research. I have great respect and admiration for his exceptional mentoring and training that helped me grow as a researcher.

I am grateful to Dr. Wei Hu for his support throughout our collaboration. I would also like to extend my sincere thanks to the members of my dissertation committee, Dr. Vassilis Athitsos, Dr. Gautam Das, Dr. Leonidas Fegaras, for their time and critical feedback.

Thanks to everyone in the IDIR lab. You all made the lab a warm environment ideal for working. I could not have wished for better colleagues. Special thanks to Fatma, the most friendly, affectionate person who planned all the fun activities in the IDIR lab and has always been available for help. Thank you, Dora, for being such a supportive friend. Thanks to Damian for managing the lab's servers and being constantly available for my numerous questions and problems.

Xin and Chunhai were great office mates. They are my terrific friends, and I'm thankful to them for making our office an excellent place to work.

I also want to thank Niloofar, the first friend I made when I came to Arlington. She is a genuine friend with an uplifting spirit, always standing by me through tough times and celebrating my achievements. I will always cherish our memories.

Finally, I cannot express deeply enough my gratitude for my family. They have always been the strong foundation I could rely on despite the great distance. I am forever grateful for their unconditional love, support, patience, and encouragement. My mom has always encouraged me to continue my study and valued the pursuit of higher education. This achievement would be impossible without her sacrifices. My siblings are my best friends, and I'm always thankful for their profound belief in me and for supporting me. I would also like to thank my cousin for her unlimited kindness and helping me tremendously when I first arrived in the USA.

July 29, 2021

ABSTRACT

On the Efficacy of Knowledge Graph Completion Methods, Accuracy Measures and Evaluation Protocols

Farahnaz Akrami, Ph.D.

The University of Texas at Arlington, 2021

Supervising Professor: Dr. Chengkai Li

In the active research area of employing embedding models for knowledge graph completion, particularly for the task of link prediction, most prior studies used some specific benchmark datasets to evaluate such models. Most triples in those datasets belong to reverse and duplicate relations, which exhibit high data redundancy due to semantic duplication, correlation, or data incompleteness. This is a case of excessive data leakage—a model is trained using features that otherwise would not be available when the model needs to be applied for real prediction. There are also Cartesian product relations for which every triple formed by the Cartesian product of applicable subjects and objects is a true fact. Link prediction on the aforementioned relations is easy and can be achieved with even better accuracy using straightforward rules instead of sophisticated embedding models. A more fundamental defect of these models is that the link prediction scenario, given such data, is non-existent in the real world. This dissertation thoroughly examines the impact of the aforementioned problems on the performance of different embedding models and provides a systematic evaluation of the true effectiveness of the models when the unrealistic triples are removed. Our experiment results show that the reverse triples led to a substantial overestimation of the accuracy of the embedding models. We argue that the popular benchmark datasets are entirely misleading and should not be used anymore.

In this dissertation, we also demonstrate the inadequacy of existing evaluation metrics. The frequently used metrics are based on the closed-world assumption and thus have flaws when a model correctly predicts a triple that does not exist in the benchmark dataset. Models are penalized for generating such correct predictions, which contradicts with the exact goal of link prediction—finding correct triples that do not already exist in the knowledge graph. Another limitation of the current metrics is that they aggregate the predictions' accuracy of all triples into a single value. This makes it impossible to discern the specific strengths and weaknesses of the models for different predictions tasks. We present the results per relation for various models and the results on relations with varying difficulty levels (1-1 vs. 1-N vs. N-M), in order to provide more insights into embedding models and show performance differences that global metrics would not reveal.

Link prediction task, the most generic evaluation protocol, verifies that models prioritize correct answers over wrong ones for a question that is already known to have an answer. This evaluation setup can be misleading as we cannot verify if a model ranks false or nonsensical triples lower than correct triples. Hence, we report the results of an alternative protocol called entity-pair ranking to rank all possible triples for a specific relation. Also, the current evaluation protocol is based on the assumption that the presence of a particular property on an entity is already known. The evaluation focuses on whether a model can derive the correct property values. In reality, though, it remains a challenge to determine whether a property is valid for a given entity in the first place. Therefore, we propose the property prediction task as another evaluation protocol. The performances of the models on these two tasks are unsatisfactory and differ considerably from link prediction results. Another way of evaluation is to find the models' performance on triple classification. This task is the binary classification of triples regarding whether they are true or false facts. We compared the classification results using two sets of negative triples, one that complies with type constraints of Freebase and one that violates them. The results show that when negative triples are type consistent, classification performance degrades considerably. The results of these different evaluation protocols suggest that better knowledge graph embedding models or training strategies are needed.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

Introduction

Large-scale knowledge graphs such as Freebase [2], DBpedia [3], NELL [4], Wikidata [5], and YAGO [6] store real-world facts as triples in the form of (head entity (subject), *relation*, tail entity (object)), denoted (h, *r*, t), e.g., (Ludvig van Beethoven, *profession*, Composer). In a knowledge graph, nodes represent entities such as places and people, and edges represent the relation between entities. They are an important resource for many AI applications, such as question answering [7, 8, 9], search [10], recommender systems [11] and smart healthcare [12], to name just a few. Despite their large sizes, knowledge graphs are far from complete in most cases, which hampers their usefulness in these applications.

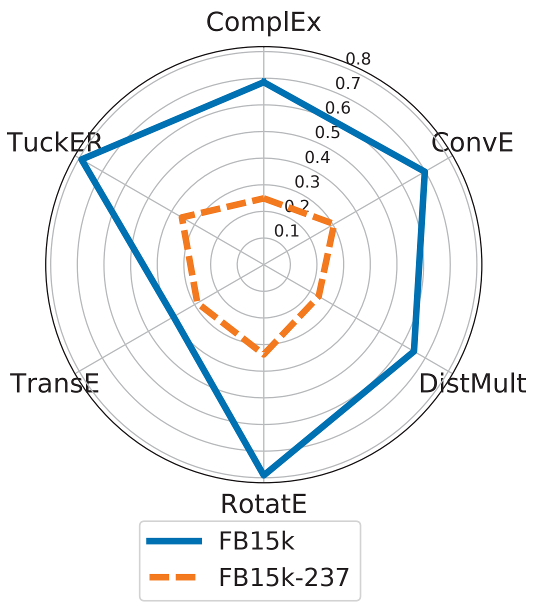
To address this important challenge, various methods have been proposed to automatically complete knowledge graphs. Existing methods in this active area of research can be categorized into two groups [13]. One group is based on *latent feature models*, also known as *embedding models*, including TransE [14], RESCAL [15], and many other methods [16, 17]. The other group is based on *observed feature models* that exploit observable properties of a knowledge graph. Examples of such methods include rule mining systems [18] and path ranking algorithms [19].

Particularly, the latent feature models are extensively studied. They embed each entity h (or t) into a multi-dimensional vector $\mathbf{h}$ (or $\mathbf{t}$). A relation r can have different representations. For example, in RESACL [15], each relation is a weight matrix whose entries specify the interaction of latent features. In TransE [14], a relation is a vector $\mathbf{r}$ that represents a geometric transformation between the head and tail entities in the embedding space and embeddings are learned in such a way that, if (h, r, t) holds, then $\mathbf{h}+\mathbf{r} \approx \mathbf{t}$.

Embedding models have been extensively evaluated on *link prediction*, a task that predicts the missing h in triple (?, r, t) or missing t in (h, r, ?). Two benchmark datasets FB15k (a subset of Freebase) and WN18 (extracted from WordNet [20]), created by Bordes et al. [14], are almost always used in such evaluation. Toutanova and Chen [21] noted that FB15k contains many reverse triples, i.e., it includes many pairs of (h, r, t) and (t, r^{-1}, h) where r and r^{-1} are reverse relations. They constructed another dataset, FB15k-237, by only keeping one relation out of any pair of reverse relations. Similarly, Dettmers et al. [22] created WN18RR out of WN18 by removing reverse triples. The community has started to use FB15k-237 and WN18RR in evaluating models and noted significant performance degeneration of existing models in comparison with their performance on FB15k and WN18 [22, 21, 23, 24].

Overestimated accuracy of embedding models due to reverse relations

This dissertation thoroughly examines the impact of reverse triples in FB15k and WN18 (details in Section 3.2.1). The idiosyncrasies of the link prediction task on such data can be summarized as follows. A1) *Link prediction becomes much easier on a triple if its reverse triple is available.* A2) *For reverse triples, a straightforward method could be even more effective than complex machine learning models.* We discovered that 70% of the triples in the training set of FB15k form reverse pairs. Similarly, for 70% of the triples in its test set, reverse triples exist in the training set. For WN18, these two percentages are even higher—92.5% and 93%. The abundant reverse triples suggest that embedding models would have been biased toward learning whether two relations r_1 and r_2 form a reverse pair. Instead of complex models, one may achieve this goal by using statistics of the triples to derive simple rules of the form (h, r_1, t) $\Rightarrow$ (t, r_2, h). In fact, we generated such a simple model which attained 71.6% for FB15k and 96.4% for WN18 using `FHits@1`$^{\uparrow}$, a common



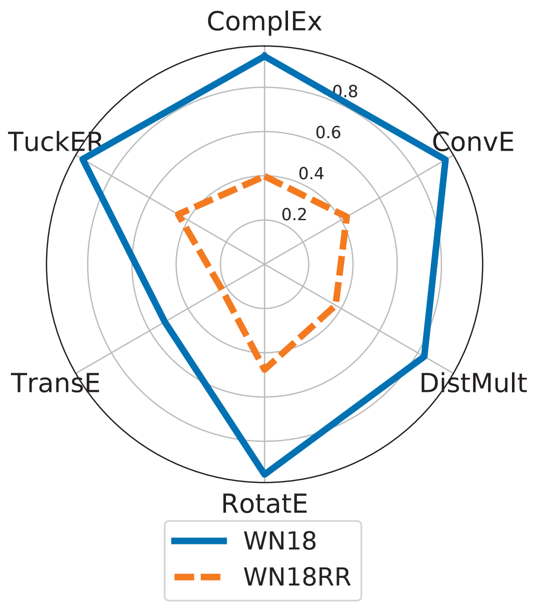


Figure 1.1: Performance of embedding models on FB15k vs. FB15k-237 and WN18 vs. WN18RR using $\mathtt{FMRR}^{\uparrow}$

accuracy measure for embedding models (more information on accuracy measures can be found in Section 2.3.2). [1] These results are on par with those by the best performing embedding models—73.8% and 94.6% on FB15K and WN18, respectively, as can be seen from Table 3.12 in Section 3.4.

The above analysis suggests that *the reverse triples led to a substantial over-estimation of the embedding models' accuracy*, which is verified by our experiments on a wide range of models. While Section 3.4 examines the results in detail, Figure 1.1 illustrates the performance comparison of a few representative models using another popular measure $\mathtt{FMRR}^{\uparrow}$. The results show that R1) *the performance of all existing embedding models degenerates significantly after reverse triples are removed*. R2) *Many successors of the original TransE*

[1] An upward/downward arrow beside a measure indicates that methods with greater/smaller values by that measure possess higher accuracy.

model were empirically shown to outperform TransE by far on FB15k, but they only attained similar or even worse performance on FB15k-237. For example, the `FHits@10`$^{\uparrow}$ of ComplEX vs. TransE is 42.3% vs. 47.5% on FB15k-237, in stark contrast to 83.2% vs. 62.4% on FB15k. R3) *The absolute accuracy of all models is poor, rendering them ineffective for real-world link prediction task.* For example, TuckER [25] attains the best `FMRR`$^{\uparrow}$ on FB15k-237 (0.355). However, its performance on FB15k (0.79) was considerably stronger. Similarly, RotatE [26] has 0.95 `FMRR`$^{\uparrow}$ on WN18 but only 0.476 on WN18RR.

The existence of excessive reverse triples in FB15k and WN18—the de facto benchmark datasets for link prediction—actually presents a more fundamental defect in many of these models: A3) *the link prediction scenario, given such data, is non-existent in the real-world at all.* With regard to FB15k, the redundant reverse relations, coming from Freebase, were just artificially created. When a new fact was added into Freebase, it would be added as a pair of reverse triples, denoted explicitly by a special relation *reverse_property* [27, 28]. In WN18, 17 out of the 18 relations are reverse relations. Some are reverse of each other, e.g., *hypernym* and *hyponym*—flower is a hypernym of sunflower and sunflower is a hyponym of flower. Others are self-reciprocal, i.e., symmetric relations such as *verb_group*—(begin, *verb_group*, start) and (start, *verb_group*, begin) are both valid triples. For such intrinsically reverse relations that always come in pair when the triples are curated into the datasets, there is not a scenario in which one needs to predict a triple while its reverse is already in the knowledge graph. Training a knowledge graph completion model using FB15k and WN18 is thus a form of *overfitting* in that the learned model is optimized for the reverse triples which cannot be generalized to realistic settings. More precisely, this is a case of excessive *data leakage*—the model is trained using features that otherwise would not be available when the model needs to be applied for real prediction. There could be more natural reverse triples that are worth prediction—two relations are not semantically reverse but correlate and/or the reverse triples are not available together in the knowledge graph due to how the data are collected. We discuss such cases of data redundancy below.

Overestimated accuracy of embedding model due to other data redundancy and Cartesian product relations

The data leakage due to reverse triples is a form of data redundancy that unrealistically inflates the models' accuracy. We identified other types of data redundancy in FB15k and another evaluation dataset YAGO3-10 (Section 3.2.2). Specifically, some relations are *duplicate* as their subject-object pairs substantially overlap, and some are *reverse duplicate* when one relation's subject-object pairs overlap a lot with another relation's object-subject pairs.

We also discovered another type of relations, which we call *Cartesian product relations* (Section 3.3), that unrealistically inflate a model's accuracy. Given such a relation, there are a set of subjects and a set of objects, and the relation is valid from every subject in the first set to every object in the second set. In a May 2013 snapshot of Freebase, close to 10% of the relations are Cartesian product relations. In FB15k, 142 out of the 1345 relations are such relations. One example is *position*, since every team in a certain professional sports league has the same set of positions. The link prediction problem for such relations thus becomes predicting, say, whether an NFL team has the quarterback position, which is not very meaningful in the real-world. Moreover, when a substantial subset of the aforementioned subject-object Cartesian product is available in the training set, it is relatively easy for a model to attain a strong prediction accuracy.

The aforementioned analyses A1-A3 on reverse relations are also applicable on duplicate and Cartersian product relations, and similarly the observations R1-R3 can be made from our experiment results. A1) In evaluating prediction models, it is misleading to mix such straightforward relations with more realistic, challenging relations. In the test set of FB15k, the numbers of reverse relations, duplicate and reverse duplicate relations, Cartesian product relations, and the remaining relations are 798, 118, 78, and 106, respectively. The `FMRR`$^\uparrow$ of ConvE on such relations is 0.72, 0.948, 0.881, and 0.444, respectively. Another example is YAGO3-10 which has two largely duplicate relations *isAffiliatedTo* and *playsFor* that account for more than 63% of its test set. The `FMRR`$^\uparrow$ of RotatE [26] is 0.612 on these 2 relations but only 0.304 on other relations. A2) Instead of learning complex embedding models, a simpler approach can be more effective. For duplicate and reverse duplicate relations, a simple rule based on data statistics can already be quite accurate, as similarly in the aforementioned case of reverse relations. For Cartesian product relations, by observing that a large percentage of possible subject-object pairs in a relation exist in the dataset, one can derive the relation is a Cartesian product relation and thus the same relation should exist in all such pairs. Our experiments on 9 Cartesian product relations in FB15k obtained an average `FHits@10`$^\uparrow$ of 98.3% using this method, which is higher than the 96.3% `FHits@10`$^\uparrow$ of TransE on these relations. A3) The existence of Cartesian product relations in FB15k is quite artificial. In fact, 60% of them are due to special "mediator nodes" in Freebase that represent multiary relationships [27] (details in Section 3.1.1) and simplification in FB15k for removing such nodes through concatenating edges. Similarly, a vast majority of the duplicate and reverse duplicate relations in FB15k were artificially created. The dataset has 84 pairs of duplicate relations. In 80 out of the 84 pairs, one or both relations are concatenated. The numbers are 63 out of 67 pairs for reverse duplicate relations. Just like reverse triples, they render a link prediction scenario largely nonexistent in the real-world and lead to unrealistically strong prediction accuracy.

Call for reinvestigation of knowledge graph completion methods and evaluation datasets

The much weaker performance of embedding models on FB15k-237 and WN18RR also drove us to examine observed feature models, specifically using rules discovered by the rule mining system AMIE [18]. Our experiment results show that it also degenerates significantly on the more realistic FB15k-237 and WN18RR. Its $\texttt{FMRR}^{\uparrow}$ on FB15k vs. FB15k-237 is 0.797 vs. 0.308 and is 0.94 vs. 0.357 on WN18 vs. WN18RR.

The embedding models generate a ranked list of candidate predictions which can be as long as the number of entities in a knowledge graph. For this ranked list to effectively assist human curators in completing the knowledge graph, the correct predictions should be ranked high. From this perspective, this dissertation depicts a realistic picture of existing methods being much less accurate than one may perceive. As mentioned in R3, their absolute accuracy is poor, which renders link prediction a task without truly effective automated solution. Hence, we call for re-investigation of possible effective approaches to completing knowledge graphs.

This dissertation presents a systematic study with the main objective of assessing the true effectiveness of link prediction methods in real-world settings. Other studies continue to evaluate models using both FB15k and FB15k-237 (similarly WN18 and WN18RR), merely viewing the latter as a more challenging dataset. However, based on our analyses A1-A3 and experiment results R1-R3, *we argue that FB15k and WN18 are completely misleading and should not be used anymore*. Similarly, our results show that YAGO3-10, which has been recently used in some studies [22], also suffers from the same defect since the majority of its triples are duplicates.

Inadequacy of existing metrics for evaluating link prediction models

The goal of link prediction is to rank high new correct triples that are missing from an existing knowledge graph. But evaluation metrics that are commonly found in prior studies penalize models for exactly that—a model's evaluation score is lowered if it produces rather correct predictions that are not found in the labeled dataset, i.e., the knowledge graph itself. The embedding models generate ranked lists of candidate predictions, of which the accuracy is measured using ranking-based information retrieval metrics such as Mean Rank (MR) and Mean Reciprocal Rank (MRR) (details in Section 2.3.2). The knowledge graph is divided into a training set and a test set. The evaluation focuses on whether a model can correctly predict the triples in the test set by generalizing from the training set. This evaluation approach assumes a *closed world*—any prediction that is not already in the knowledge graph is considered wrong. On the contrary, the open-world assumption states that failure to derive a fact would not imply its opposite [29]. This assumption is more suitable for modeling incomplete knowledge such as the benchmark datasets FB15k and FB15k-237. This is verified by our experiments on FB15k-237, a link prediction is identified as correct if it exists in a May 2013 snapshot of Freebase even if it is missing from the FB15k-237 test set. This way of evaluation led to improved model accuracy measures, proving that the closed-world assumption underestimates model performance.

Another problem with the commonly used evaluation metrics is that they aggregate a model's prediction performance on all relations and all triples into a single accuracy value. This aggregation makes it impossible to discern the model's strengths and weaknesses on different types of prediction tasks separately. In section 3.4.3, we present some detailed results of models. These results provide more insights into models. For example, we can see that models have high performance on symmetric relations of WN18RR or that on 1-1 relations of FB15k-237, RotatE has the highest performance while ConvE has the lowest performance. We also show that FB15k-237 is highly skewed with a few relations of high frequency and a large number of relations with low frequency. Hence, we suggest using the macro average of $\texttt{FMRR}^{\uparrow}$ per relation to treat all the relations equally.

Inadequacy of existing tasks and protocols for evaluating link prediction models

Link prediction, the most generic evaluation protocol used to gauge the performance of knowledge graph embedding models, verifies that models prioritize correct answers over wrong ones for a question that is already known to have an answer. Wang et al. [1] discussed that this evaluation setup could be misleading because it does not verify whether all false and nonsensical triples are ranked lower than correct triples. They proposed an alternative protocol called *entity-pair ranking* to rank all possible triples for a specific relation. We reported the performance of the models based on this protocol, and our results verify that some nonsensical triples are ranked higher than correct ones. For example, on relation *continents/countries_within* which shows the relation between a continent and its countries, we observe that false (subject, object) entity pairs such as (2000 Summer Olympics ,The London 2012 Summer Olympics) and (Lycoming County, Williamsport) are ranked higher than a correct pair such as (Europe, Spain). Further, the current evaluation protocol assumes that the presence of a particular property on an entity is already known and thus the evaluation focuses on whether a model can derive the correct property values. In reality, though, it remains a challenge to determine whether a property is valid for a given entity in the first place. Therefore, we propose the property prediction task as another evaluation protocol. The performance of the models on this task is unsatisfactory, and the highest $\texttt{FMRR}^{\uparrow}$ is 0.182. Another evaluation protocol is to find the performance of the models on the triple classification task, determining if a triple is true or false. To conduct this task, some negative triples should be included into test and validation sets. Previous studies usually generated negative triples by randomly replacing a triple's head or tail entity with another entity. Triple classification on these randomly created negative samples is trivial, and the models have strong performance [30, 31]. We created negative samples by replacing head/tail entities with other type-consistent entities to have more challenging negative triples. We compared the classification results of these samples with another set of type-violating samples. Given a test triple (h, *r*, t), a type-consistent negative sample could be created by replacing h/t with another entity h'/t' that has the same type as h/t. To create a type-violating negative example, type of h'/t' should be different from h/t. As an example, consider the test triple is (Carl Foreman, *place_of_birth*, Chicago) then the type-consistent sample will be (Carl Foreman, *place_of_birth*, Los Angeles) and the type-violating one will be (Carl Foreman, *place_of _birth*, 55th Primetime Emmy Awards). The results show that, when negative triples are type consistent, classification performance degrades considerably. All these different evaluation protocols together provide more robust assessment of models' true efficacy in reality. The evaluation results using these protocols suggest that better knowledge graph completion methods and training strategies are needed.

The structure of this dissertation is as follows:

- Chapter 2 provides a survey of some important embedding models as well as the observed feature model AMIE. We also present a detailed explanation of evaluation datasets and measures used to find the performance of the models.
- Chapter 3 provides a thorough investigation of the data redundancy problem in how existing embedding models for knowledge graph completion were trained, due to reverse and duplicate triples in the de facto benchmark datasets FB15k and WN18. We also explain the existence of Cartesian product relations in FB15k. The results of a comprehensive evaluation of these defects' impacts on the performance of many representative embedding models are presented in this chapter. The results of link prediction using the observed feature model AMIE are also provided in Chapter 3.
- Chapter 4 discusses the inadequacy of evaluation metrics. We provide the experiments' results to demonstrate how the identified issues impact the models' performance.
- In Chapter 5, we discuss the inadequacy of the commonly used evaluation protocol, the link prediction task, in gauging models' performance. We discuss other possible evaluation protocols, including entity-pair ranking, property prediction and triple classification, and present the results using these protocols.
- We conclude this dissertation in Chapter 6.

CHAPTER 2

Background: Knowledge Graph Completion

This chapter briefly summarizes representative knowledge graph completion methods. Existing methods can be categorized into two groups [13], including *latent feature models*, also known as *embedding models*, and *observed feature models* that exploit observable properties of a knowledge graph. In our description, vectors are represented as bold lower case letters such as $\mathbf{x}$. $[\mathbf{x}]_i$ represents the ith element of $\mathbf{x}$. A matrix is denoted by a bold upper case letter, e.g., $\mathbf{M}$. A knowledge graph $\mathcal{G} = \{(\mathsf{h}, r, \mathsf{t}) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}\}$ consists of a set of entities $\mathcal{E}$ and a set of relations $\mathcal{R}$. Triples are represented as (h, r, t) where h, $\mathsf{t} \in \mathcal{E}$ are the head and tail entities, and relationship $r \in \mathcal{R}$ exists from the head to the tail. $\langle \mathbf{x}, \mathbf{y}, \mathbf{z} \rangle = \sum_i [\mathbf{x}]_i \cdot [\mathbf{y}]_i \cdot [\mathbf{z}]_i$ is the component-wise multi-linear dot product.

2.1 Latent Feature Models

In this section, we introduce a family of state-of-the-art embedding-based models that were used in our experiments. Embedding-based methods employ two crucial components: (1) a scoring function to measure the plausibility of triples (h, r, t), and (2) a process to learn the representations (i.e., embeddings) of entities and relations by solving an optimization problem of maximizing the scores of correct triples while minimizing the scores of incorrect ones.

Table 2.1: Scoring functions of embedding models

Model	**Scoring function** $f_r(\mathsf{h},\mathsf{t})$
TransE [14]	$-\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|^2_{\ell_1/\ell_2}$
TransH [31]	$-\|(\mathbf{h}-\mathbf{v}_r^\top\mathbf{h}\mathbf{v}_r)+\mathbf{d}_r-(\mathbf{t}-\mathbf{v}_r^\top\mathbf{t}\mathbf{v}_r)\|$
TransR [32]	$-\|\mathbf{M}_r\mathbf{h}+\mathbf{r}-\mathbf{M}_r\mathbf{t}\|^2_{\ell_2}$
TransD [33]	$-\|(\mathbf{r}_p\mathbf{h}_p^\top+\mathbf{I})\mathbf{h}+\mathbf{r}+(\mathbf{r}_p\mathbf{t}_p^\top+\mathbf{I})\mathbf{t}\|$
RotatE [26]	$-\|\mathbf{h}\circ\mathbf{r}-\mathbf{t}\|^2$
RESCAL [15]	$\mathbf{h}^\top\mathbf{W}_r\mathbf{t}$
DistMult [34]	$\langle\mathbf{h},\mathbf{w}_r,\mathbf{t}\rangle$
ComplEx [35]	$\mathrm{Re}(\langle\mathbf{h},\mathbf{w}_r,\mathbf{t}\rangle)$
TuckER [25]	$\mathcal{W}\times_1\mathbf{h}\times_2\mathbf{r}\times_3\mathbf{t}$
ConvE [22]	$g(vec(g([\mathbf{M}_h;\mathbf{M}_r]*\omega))\mathbf{W})\mathbf{t}$

2.1.1 Scoring Function

A scoring function $f:\mathcal{E}\times\mathcal{R}\times\mathcal{E}\to\mathbb{R}$ assigns a real value score to the triple (h, r, t) given the relation and entity embeddings. Table 2.1 summarizes the scoring functions of the embedding models discussed in this section.

TransE: In TransE [14], the scoring function is as follows.

$$f_r(\mathsf{h},\mathsf{t})=-\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|^2_{\ell_1/\ell_2} \tag{2.1}$$

where ℓ_1, ℓ_2 represent L1 and L2 norms, respectively, and $\mathbf{h},\mathbf{t},\mathbf{r}\in\mathbb{R}^d$. In TransE, a relation is a vector $\mathbf{r}$ that represents a geometric transformation between the head and tail entities in the embedding space and embeddings are learned in such a way that, if (h, r, t) holds, then $\mathbf{h}+\mathbf{r} \approx \mathbf{t}$. TransE is a scalable method with a small number of model parameters, but it has limitations in modeling 1-to-n, n-to-1, and m-to-n relations [31].

TransH: TransH [31] is similar to TransE but it aims to address TransE's limitations by not using the same embedding for an entity in different relations. Consider the following example. In principle the films produced by Universal Studios should be represented as similar vectors when we focus on a relation about film production. When the relation in focus becomes film genre instead, the vectors for films in different genres should be far from each other. Albeit intuitive, such is not possible in TransE. TransH models each relation as a vector $\mathbf{d}_r$ on a relation-specific hyperplane $\mathbf{v}_r$ (i.e., the normal vector where $\|\mathbf{v}_r\| = 1$), and the entity embeddings $\mathbf{h}$ and $\mathbf{t}$ are projected to this hyperplane to obtain:

$$\mathbf{h}_\perp = \mathbf{h} - \mathbf{v}_r^\top \mathbf{h} \mathbf{v}_r \tag{2.2}$$

$$\mathbf{t}_\perp = \mathbf{t} - \mathbf{v}_r^\top \mathbf{t} \mathbf{v}_r \tag{2.3}$$

In this way, the embeddings of entities are learned differently for each relation. Then, the projected embeddings are used to calculate the score of the triple:

$$f_r(\mathsf{h},\mathsf{t}) = -\|\mathbf{h}_\perp + \mathbf{d}_r - \mathbf{t}_\perp\| \tag{2.4}$$

TransR: Lin et al. [32] proposed TransR which learns the embeddings in two different vector spaces $\mathbb{R}^d$ and $\mathbb{R}^k$ for entities and relations, where the dimensions k and d are not necessarily identical. They argued that using the same semantic space for entities and relations, as in TransE and TransH, is insufficient because they are two completely different types of objects. Instead, TransR defines a projection matrix $\mathbf{M}_r \in \mathbb{R}^{k \times d}$ to map entity embeddings to the vector space for each relation.

$$\mathbf{h}_\perp = \mathbf{M}_r \mathbf{h} \tag{2.5}$$

$$\mathbf{t}_\perp = \mathbf{M}_r \mathbf{t} \mathbf{v}_r \tag{2.6}$$

By this approach, similar entities near each other in the entity space may become far apart after being mapped to the relation space because they are different regarding particular aspects pertinent to the relation. The scoring function of TransR is defined as

$$f_r(\mathsf{h},\mathsf{t}) = -\|\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp\|_{\ell_2}^2 \tag{2.7}$$

TransD: In TransD [33], which improves over TransR, the entities and the relation in a triple (h, r, t) are represented by two groups of vectors: one being $\mathbf{h}$, $\mathbf{r}$ and $\mathbf{t}$, and the other being $\mathbf{h}_p$, $\mathbf{r}_p$ and $\mathbf{t}_p$. The latter group defines the projection matrices as follows.

$$\mathbf{M}_{rh} = \mathbf{r}_p\mathbf{h}_p^\top + \mathbf{I} \tag{2.8}$$

$$\mathbf{M}_{rt} = \mathbf{r}_p\mathbf{t}_p^\top + \mathbf{I} \tag{2.9}$$

As in TransR, these projection matrices are used to map entity vectors $\mathbf{h}$ and $\mathbf{t}$ to the relation vector space. However, in contrast to TransR, there is a unique projection matrix for each entity-relation pair. The argument given by [33] is that different types of entities connected to a relation should have different mapping matrices and one projection matrix per relation, as in TransR, is insufficient. Furthermore, computationally expensive matrix-vector multiplication in TransR poses a challenge in scaling it up. TransD tackles this challenge by using vector operations instead.

RotatE: RotatE [26] defines each relation as a rotation from the source entity to the target entity in the complex vector space. The scoring function is as follows,

$$f_r(\mathsf{h},\mathsf{t}) = -\|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|^2 \tag{2.10}$$

where $\mathbf{h}, \mathbf{r}, \mathbf{r} \in \mathbb{C}^d$, $|r_i| = 1$, and $\circ$ is the Hadamard (or element-wise) product. In this way the approach aims to effectively model symmetric or antisymmetric, inversion, and composition relation patterns [26].

Another group of embedding approaches formulate link prediction as a third-order binary tensor completion problem in which a knowledge graph is represented as a partially observed tensor $\mathbf{Y} \in \{0,1\}^{|\mathcal{E}|\times|\mathcal{E}|\times|\mathcal{R}|}$. An entry in $\mathbf{Y}$ equals one if the corresponding triple exists in $\mathcal{G}$. Different models such as RESCAL [15], DistMult [34], ComplEx [35], and TuckER [25] used various methods of tensor factorization to decompose $\mathbf{Y}$ and assign scores to triples based on the learned factors.

RESCAL: RESCAL [15] is a collective matrix factorization model which represents a relation as a matrix $\mathbf{W}_r \in \mathbb{R}^{d\times d}$ that describes the interactions between latent representations of entities. The score of a triple in this method is defined as:

$$f_r(\mathsf{h},\mathsf{t}) = \mathbf{h}^\top \mathbf{W}_r \mathbf{t} = \sum_{i=1}^{d}\sum_{j=1}^{d} w_{ij}^{(r)} h_i t_j \tag{2.11}$$

DistMult: DistMult [34] is similar to RESCAL but it restricts relations to be diagonal matrices $\mathbf{w}_r \in \mathbb{R}^d$ in order to reduce the number of relation parameters:

$$f_r(\mathsf{h},\mathsf{t}) = \langle \mathbf{h}, \mathbf{w}_r, \mathbf{t} \rangle \tag{2.12}$$

Due to this simplification, DistMult can only model symmetric relations.

ComplEx: ComplEx [35] is an extension of DistMult. It uses complex numbers instead of real numbers to handle symmetric and anti-symmetric relations.

$$\begin{aligned} f_r(\mathsf{h},\mathsf{t}) &= Re(\langle \mathbf{h}, \mathbf{w}_r, \mathbf{t} \rangle) \\ &= \langle Re(\mathbf{h}), Re(\mathbf{w}_r), Re(\mathbf{t}) \rangle \\ &\quad + \langle Im(\mathbf{h}), Re(\mathbf{w}_r), Im(\mathbf{t}) \rangle \\ &\quad + \langle Re(\mathbf{h}), Re(\mathbf{w}_r), Im(\mathbf{t}) \rangle \\ &\quad - \langle Im(\mathbf{h}), Im(\mathbf{w}_r), Im(\mathbf{t}) \rangle \end{aligned} \tag{2.13}$$

where $\mathbf{h}, \mathbf{t}, \mathbf{w}_r \in \mathbb{C}^d$ and $Re(\mathbf{x})$, $Im(\mathbf{x})$ denote the real and imaginary vector components of vector $\mathbf{x}$.

TuckER: TuckER [25] is a model based on the Tucker decomposition [36] of the binary tensor of triples $\mathbf{Y}$. Tucker decomposition factorizes a given tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ into a set of matrices and a smaller so-called core tensor:

$$\mathcal{X} \approx \mathcal{T} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} \tag{2.14}$$

where $\mathcal{T} \in \mathbb{R}^{P \times Q \times R}$, $\mathbf{A} \in \mathbb{R}^{I \times P}$, $\mathbf{B} \in \mathbb{R}^{J \times Q}$, and $\mathbf{C} \in \mathbb{R}^{K \times R}$. $\times_n$ indicates the tensor product and n shows along which mode the product is computed. $\mathcal{T}$ is the core tensor and elements of it express to which extend different tensor elements interact. In TuckER, binary tensor of triples is factorized with entity embedding matrix $\mathbf{E} = \mathbf{A} = \mathbf{C} \in \mathbb{R}^{|\mathcal{E}| \times d}$ and relation embedding matrix $\mathbf{R} = \mathbf{B} \in \mathbb{R}^{|\mathcal{R}| \times k}$. The scoring function of this model is defined as:

$$f_r(\mathsf{h}, \mathsf{t}) = \mathcal{W} \times_1 \mathbf{h} \times_2 \mathbf{r} \times_3 \mathbf{t} \tag{2.15}$$

where $\mathbf{h}$ and $\mathbf{t} \in \mathbb{R}^d$ are rows of $\mathbf{E}$ and $\mathbf{r} \in \mathbb{R}^k$ is a row of $\mathbf{R}$. $\mathcal{W} \in \mathbb{R}^{d \times k \times d}$ is the core tensor.

ConvE: ConvE [22] is a neural network model that uses 2D convolutional layers over embeddings, and interactions between entities and relations are modeled by convolutional and fully connected layers. For a triple (h, r, t), the vectors $\mathbf{h}$, $\mathbf{r} \in \mathbb{R}^d$ are reshaped into 2D matrices $\mathbf{M}_h$, $\mathbf{M}_r \in \mathbb{R}^{d_m \times d_n}$, where $d = d_m \times d_n$. $\mathbf{M}_h$ and $\mathbf{M}_r$ are then concatenated and used as the input of a 2D convolutional layer with filters ω. The score of the triple is calculated as:

$$f_r(\mathsf{h}, \mathsf{t}) = g(vec(g([\mathbf{M}_h; \mathbf{M}_r] * \omega))\mathbf{W})\mathbf{t} \tag{2.16}$$

where g is a generic activation function, and vec is the vectorization operation reshaping a tensor into a vector.

2.1.2 Loss Function

To learn entity and relation representations, the embeddings are usually initialized randomly. Then through the training process a loss function is minimized—embeddings are optimized such that the scores of positive triples are maximized while the scores of negative ones are minimized. Since a real-world knowledge graph contains only positive triples, negative triples are generated by *corrupting* the positive triples in the training set—a process that replaces the head or tail entity of each positive triple by other entities in the knowledge graph [14, 31]. There are different ways to generate negative triples such as uniform negative sampling and Bernoulli negative sampling [17].

In the following, some of the frequently used loss functions in embedding models are briefly explained. Pointwise approaches capture the difference between triples' scores and their true labels, while in pairwise approaches the loss is defined as the difference between positive and negative triples' scores [37].

Pointwise square error loss: In this loss function, the objective is to minimize the squared difference between triples' scores and their true labels $y_{hrt} \in \{-1, 1\}$ (+1/-1 for positive/negative example):

$$\mathcal{L} = \frac{1}{2} \sum_{\mathsf{h},\mathsf{t} \in \mathcal{E}, r \in \mathcal{R}} (f_r(h,t) - y_{hrt})^2 \tag{2.17}$$

This loss function is used in RESCAL [15].

Pointwise logistic loss: In this loss function, a logistic function is used to maximize the scores of positive triples while minimizing the scores of negative ones, as follows.

$$\mathcal{L} = \sum_{(\mathsf{h},r,\mathsf{t}) \in S \cup S'} log(1 + exp(-y_{hrt} \times f_r(\mathsf{h},\mathsf{t}))) \tag{2.18}$$

where S is the set of positive triples (h, r, t) in the training set, and S' is the set of negative triples (h', r, t'). Logistic loss is used to train ComplEx [35].

Pairwise margin-based loss: TransE [14] and DistMult [34] models are learned by a margin-based ranking objective.

$$\mathcal{L} = \sum_{(\mathsf{h},r,\mathsf{t}) \in S} \sum_{(\mathsf{h}',r,\mathsf{t}') \in S'} max(0, f_r(\mathsf{h},\mathsf{t}) + \gamma - f_r(\mathsf{h}',\mathsf{t}')) \tag{2.19}$$

where γ is the margin that separates positive triples from negative ones.

2.2 Observed Feature Models

In contrast to embedding models which employ latent features of knowledge graphs, observed feature models directly exploit observable features. For example, one can derive the ancestor-descendant relationship between two persons captured by the rule (a, *father_of*, b) $\wedge$ (b, *father_of*, c) $\Rightarrow$ (a, *ancestor*, c) which can be also represented as *father_of*(a,b) $\wedge$ *father _of*(b,c) $\Rightarrow$ *ancestor*(a,c). A representative of the observed feature models is the rule mining system AMIE [18]. Compared with the embedding-based methods, rule-mining systems have the advantage of being more interpretable.

AMIE mines Horn rules in the aforementioned form. Mining Horn clauses from data has long been extensively studied in the field of *inductive logic programming* (ILP) [38]. However, methods proposed in this field are not applicable to knowledge graphs due to two reasons. First, ILP systems are not efficient enough to cope with large knowledge graphs. Second, in addition to positive examples, they also rely on negative examples which are non-trivial to come by as general knowledge graphs explicitly record only known facts. Under the *open-world assumption*, which is far more realistic than the *closed-world assumption* [29] in the context of knowledge graphs, facts nonexistent or cannot be proven based on available knowledge are not necessarily *false* and can be just interpreted as *unknown*. In fact, the Web Ontology Language (OWL) follows the open-world assumption [39]. AMIE tackles these challenges with an efficient rule mining algorithm and a method to generate negative examples using *partial completeness assumption*.

In AMIE, a rule has a body (antecedent) and a head (consequent), represented as $B_1 \wedge B_2 \wedge \ldots \wedge B_n \Rightarrow H$ or in simplified form $\vec{B} \Rightarrow H$. The body consists of multiple *atoms* $B_1, \ldots, B_n$ and the head H itself is also an atom. In an atom *r*(h,t), which is another representation of fact triple (h, *r*, t), the subject and/or the object are variables to be instantiated.

The prediction of the head can be carried out when all the body atoms can be instantiated in the knowledge graph.

One popular measure of the quality of a mined rule is *confidence*, defined as follows in which $\mathsf{z}_1, \ldots, \mathsf{z}_m$ are variables.

$$conf(\overrightarrow{B} \Rightarrow r(\mathsf{h},\mathsf{t})) := \frac{support(\overrightarrow{B} \Rightarrow r(\mathsf{h},\mathsf{t}))}{\#(\mathsf{h},\mathsf{t}) : \exists \mathsf{z}_1,\ldots,\mathsf{z}_m : \overrightarrow{B}} \tag{2.20}$$

The numerator in Equation (2.20) is the support of the rule, i.e., "the number of distinct pairs of subjects and objects in the head of all instantiations" that appear in the knowledge graph [18]. The denominator is the support of the body.

This definition is based on the closed-world assumption in which unknown facts are considered false. As mentioned earlier, this assumption is far from realistic due to the incompleteness of general real-world knowledge graphs. Hence, Galárraga et al. [18] proposed *PCA confidence* which is based on the *partial completeness assumption*. Under this assumption, if the knowledge graph contains an object t for a pair of relation r and subject h, then all the objects for that relation-subject pair are already contained in the knowledge graph. For instance, if two children c_1 and c_2 of a person p exist as *father_of*(c_1, p) and *father_of*(c_2, p), any new fact *father_of*(c, p) will be false. On the other hand, if no child of p is known according to the knowledge graph, any fact *father_of*(c, p) is considered neither true nor false. The measure PCA confidence is thus defined as

$$conf_{pca}(\overrightarrow{B} \Rightarrow r(\mathsf{h},\mathsf{t})) := \frac{support(\overrightarrow{B} \Rightarrow r(\mathsf{h},\mathsf{t}))}{\#(\mathsf{h},\mathsf{t}) : \exists \mathsf{z}_1,\ldots,\mathsf{z}_m,\mathsf{t}' : \overrightarrow{B} \wedge r(\mathsf{h},\mathsf{t}')} \tag{2.21}$$

In Equation (2.21), the denominator is the number of facts that are known to be true plus the facts $r(\mathsf{h},\mathsf{t}')$ that are considered false under the partial completeness assumption:

$$\forall t' : r(\mathsf{h},\mathsf{t}') \text{ is false} \leftarrow r(\mathsf{h},\mathsf{t}') \notin \mathcal{G} \wedge \exists \mathsf{t} \text{ s.t. } r(\mathsf{h},\mathsf{t}) \in \mathcal{G} \tag{2.22}$$

2.3 Existing Evaluation Framework

2.3.1 Evaluation Datasets

FB15k: Most embedding models have been evaluated on FB15k, a subset of Freebase generated by Bordes et al. [14]. FB15k contains only those Freebase entities that were also available in Wikipedia based on the *wiki-links* database [1] and have at least 100 appearances in Freebase. The relations included into FB15k must also have at least 100 instances. 14,951 entities and 1,345 relations satisfy these criteria, which account for 592,213 triples included into FB15k. These triples were randomly split into training, validation and test sets. Table 2.2 shows the statistics of this and other datasets.

WN18: Many embedding models have also been evaluated using WN18 [14], a knowledge graph extracted from the English lexical database WordNet [20] which defines conceptual-semantic and lexical relations between word forms or between synsets—sets of synonyms. An example triple is (presentation, *derivationally_related_form*, present).

YAGO3-10: Some embedding models were evaluated using YAGO3-10 [22], a subset of YAGO3 [40]—the multilingual extension of YAGO [6] which is derived from Wikipedia and WordNet. YAGO3-10 contains entities that are involved in at least 10 relations in YAGO3.

2.3.2 Evaluation Methods and Measures

Embedding models have been evaluated using several highly-related knowledge graph completion tasks such as triple classification [31, 30], link prediction [41], relation extraction [42, 32], and relation prediction [43]. The *link prediction* task as described in [14] is particularly widely used for evaluating different embedding methods. Its goal is to predict the missing h or t in a triple (h, r, t). For each test triple (h, r, t), the head entity h is replaced with every other entity h' $\in \mathcal{E}$ in the dataset, to form *corrupted* triples. The original test

[1] `https://code.google.com/archive/p/wiki-links/`

Table 2.2: Statistics of evaluation datasets

Dataset	**#entities**	**#relations**	**#train**	**#valid**	**#test**
FB15k	14,951	1,345	483,142	50,000	59,071
FB15k-237	14,541	237	272,115	17,535	20,046
WN18	40,943	18	141,442	5,000	5,000
WN18RR	40,943	11	86,835	3,034	3,134
YAGO3-10	123,182	37	1,079,040	5,000	5,000
YAGO3-10-DR	122,837	36	732,556	3,390	3,359

triple and its corresponding corrupted triples are ranked by their scores according to the score functions (Section 2.1) and the rank of the original test triple is denoted $rank_{\mathsf{h}}$. The same procedure is used to calculate $rank_{\mathsf{t}}$ for the tail entity `t`. A method with the ideal ranking function should rank the test triple at top.

The accuracy of different embedding models is measured using `Hits@1`$^{\uparrow}$, `Hits@10`$^{\uparrow}$, Mean Rank (`MR`$^{\downarrow}$), and Mean Reciprocal Rank (`MRR`$^{\uparrow}$), as in [14]. `Hits@k`$^{\uparrow}$ is the percentage of top k results that are correct. `MR`$^{\downarrow}$ is the mean of the test triples' ranks, defined as:

$$MR = \frac{1}{2\,|T|} \sum_{(\mathsf{h},r,\mathsf{t})\in T} (rank_{\mathsf{h}} + rank_{\mathsf{t}}) \tag{2.23}$$

in which $|T|$ is the size of the test set. `MRR`$^{\uparrow}$ is the average inverse of harmonic mean of the test triples' ranks, defined as

$$MRR = \frac{1}{2\,|T|} \sum_{(\mathsf{h},r,\mathsf{t})\in T} (\frac{1}{rank_{\mathsf{h}}} + \frac{1}{rank_{\mathsf{t}}}) \tag{2.24}$$

Besides these raw metrics, we also used their corresponding *filtered* metrics [14], denoted `FHits@1`$^{\uparrow}$, `FHits@10`$^{\uparrow}$, `FMR`$^{\downarrow}$, and `FMRR`$^{\uparrow}$, respectively. In calculating these measures, corrupted triples that are already in training, test or validation sets do not participate in ranking. In this way, a model is not penalized for ranking other correct triples higher than a test triple. For example, consider the task of predicting tail entity. Suppose the test triple is (Tim Burton, *film*, Edward Scissorhands) and the training, test, or validation set also contains another triple (Tim Burton, *film*, Alice in Wonderland). If a model ranks Alice in Wonderland higher than Edward Scissorhands, the filtered metrics will remove this film from the ranked list so that the model would not be penalized for ranking (Tim Burton, *film*, Edward Scissorhands) lower than (Tim Burton, *film*, Alice in Wonderland), both correct triples.

We note that, by definition, higher $\texttt{Hits@1}^{\uparrow}$ ($\texttt{FHits@1}^{\uparrow}$), $\texttt{Hits@10}^{\uparrow}$ ($\texttt{FHits@10}^{\uparrow}$) and $\texttt{MRR}^{\uparrow}$ ($\texttt{FMRR}^{\uparrow}$), and lower $\texttt{MR}^{\downarrow}$ ($\texttt{FMR}^{\downarrow}$) indicate better accuracy.

CHAPTER 3

Inadequacy of Benchmarks

In this chapter we investigate the existence and impact of a few types of relations in FB15k, WN18 and YAGO3-10, including reverse (and symmetric) relations, redundant relations, and Cartesian product relations. The outcome suggests that triples in these relations led to a substantial over-estimation of the accuracy of embedding models.

3.1 Identifying the Most Probable Freebase Snapshot Used for Producing FB15k

In order to understand the various defects in FB15k and their root cause, we searched for the same Freebase snapshot that was used to create FB15k. When it was active, Freebase maintained periodic snapshots, more frequent than monthly. It is unclear from [14] which snapshot was used to create FB15k.[1] To derive the most probable timestamp of the snapshot from which FB15k was produced, we compared the snapshots available at `https://commondatastorage.googleapis.com/freebase-public/` as of June 2019. Particularly, we considered the snapshots in the several months before [14] was published. We analyzed to what degree these snapshots overlap with FB15k.

[1]We had email communication with the authors of [14]. They could not remember the exact date of the Freebase snapshot used for producing FB15k.

The May 5, 2013 snapshot [2] has the largest overlap among these snapshots, as it contains 99.54% of the triples in FB15k. We thus concluded that FB15k was most likely drawn from a snapshot around May 5, 2013, which can be approximated by the snapshot on that exact date.

3.1.1 Mediator Nodes

Mediator nodes, also called *compound value type* (CVT) nodes, are used in Freebase to represent multiary relationships [27]. For example, Figure 3.1 shows several CVT nodes. The rightmost CVT node is connected to an award, two nominee nodes and a work through various relations. In the May 2013 Freebase snapshot, for many (but not all) CVT nodes, additional *concatenated edges* were created. Specifically, for such a CVT node, multiple triples were created, each a concatenation of two edges connected through the CVT node. These binary relationships partly capture the multiary relationship represented by the CVT node. For instance, the triples (Bafta Award For Best Film, *award_category/nominees*, CVT) and (CVT, *award_nomination/nominated_for*, A Room With A View) in Figure 3.1 would be concatenated to form a triple between the award and the work nominated for the award. The concatenation of two relations r_1 and r_2 is written as $r_1.r_2$. There are 54,541,700 concatenated triples in the May 2013 snapshot.

FB15k does not include any CVT nodes or edges adjacent to such nodes from Freebase. However, it kept most *concatenated edges* (or maybe all, although we have no absolute way to verify, since nowhere we can find all Freebase snapshots that have ever been created). All the 707 concatenated relations in FB15k exist in the May 2013 snapshot. Among the 592,213 triples in FB15k, 396,148 are concatenated and 394,947 of them could be found in the snapshot.

[2] `https://commondatastorage.googleapis.com/freebase-public/rdf/freebase-rdf-2013-05-05-00-00.gz`

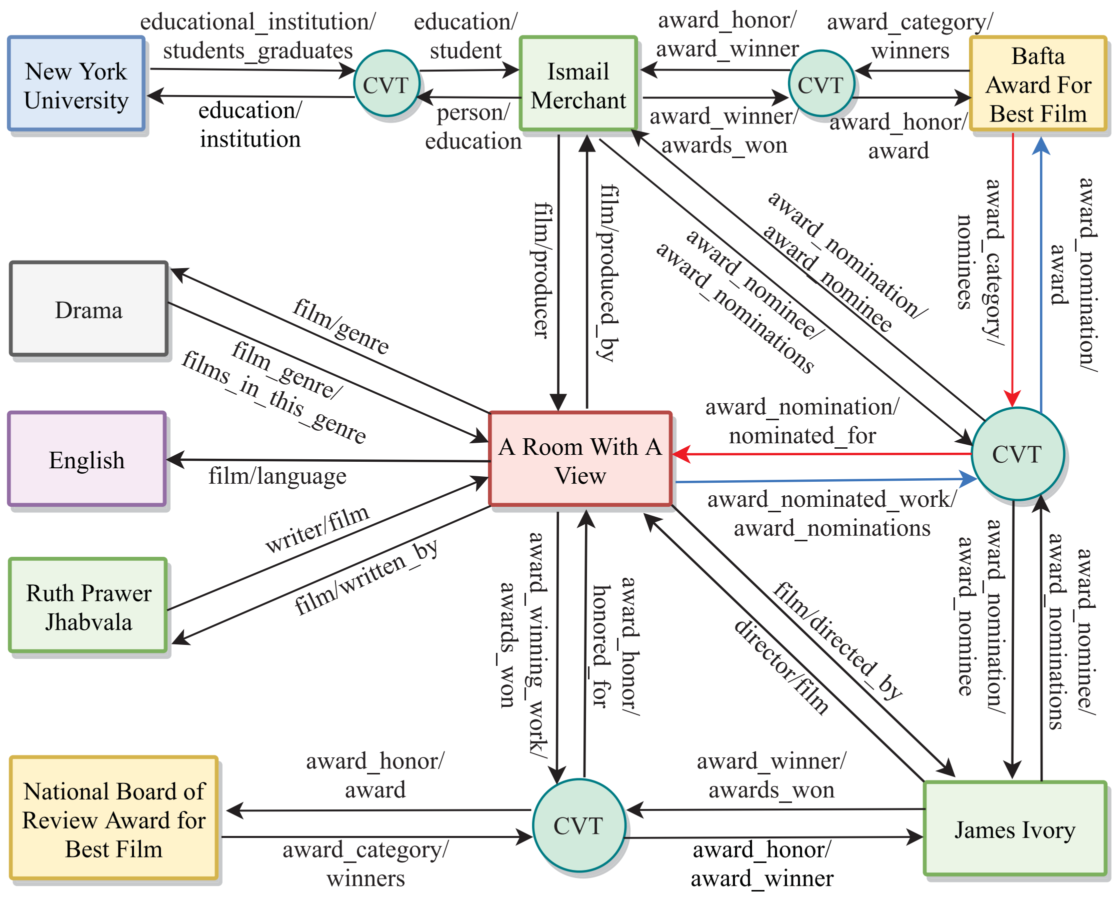


Figure 3.1: Mediator (CVT) nodes in Freebase

3.2 Data Redundancy

3.2.1 Data Leakage Due to Reverse Triples

(1) **FB15k**: In [21], Toutanova and Chen noted that the widely-used benchmark dataset FB15k contains many reverse triples, i.e., it includes many pairs of (h, r, t) and (t, r^{-1}, h) where r^{-1} is the reverse of r. Freebase actually denotes reverse relations explicitly using a special relation *reverse_property* [27, 28].[3] For instance, the triple (film/directed

[3]The relation's full name is */type/property/reverse_property*. In Freebase, the full name of a relation follows the template of /domain/entity type/relation. For instance, */tv/tv_genre/programs* represents a rela-

_by, *reverse_property*, director/film) in Freebase denotes that *film/directed_by* and *director/film* are reverse relations.[4] Therefore, (A Room With A View, *film/directed_by*, James Ivory) and (James Ivory, *director/film*, A Room With A View) form a pair of reverse triples, as shown in Figure 3.1. The reverse relation of a concatenated relation $r_1.r_2$ is the concatenation of the corresponding reverse relations, i.e., $r_2^{-1}.r_1^{-1}$. For example, in Figure 3.1 *award_nominated _work/award_nominations . award_nomination/award* (blue edges) and *award_category/nominees . award_nomination/nominated_for* (red edges) are two concatenated relations which are reverse of each other.

Using the reverse relation information from the May 2013 snapshot of Freebase, out of the 483,142 triples in the training set of FB15k, 338,340 triples form 169,170 reverse pairs. Furthermore, for 41,529 out of the 59,071 triples (i.e., about 70.3%) in the test set of FB15k, their reverse triples exist in the training set. These data characteristics suggest that embedding models would have been biased toward learning reverse relations for link prediction. More specifically, the task can largely become inferring whether two relations r_1 and r_2 form a reverse pair. Given the abundant reverse triples in the dataset, this goal could potentially be achieved without using a machine learning approach based on complex embeddings of entities and relations. Instead, one may aim at deriving simple rules of the form (h, r_1, t) $\Rightarrow$ (t, r_2, h) using statistics about the triples in the dataset. In fact, Dettmers et al. [22] generated such a simple model which attained a 68.9% accuracy by the measure FHits@1$^{\uparrow}$. We generated a similar model by finding the relations that have more then 80% intersections. It attained an FHits@1$^{\uparrow}$ of 71.6%. This is even slightly better than the 70.3% accuracy one may achieve using an oracle based on the reverse relations denoted in the

tion named *programs* belonging to domain *tv*. The subject of any instance triple of this relation belongs to type *tv_genre*. For simplicity of presentation, by default we omit the prefixes. In various places we retain the entity type to avoid confusions due to identical relation names.

[4]Note that relations *film/directed_by* and *director/film* are also special entities in (film/directed_by, *reverse_property*, director/film).

May 2013 Freebase snapshot. The `FHits@1`$^{\uparrow}$ of the best performing embedding model on FB15k is 73.8% (more details in Table 3.12 of Section 3.4).

(2) **WN18**: WN18 also suffers from data leakage, as 14 out of its 18 relations form 7 pairs of reverse relations, e.g., (europe, *has_part*, republic_of_estonia) and (republic_of_estonia, *part_of*, europe) are two reverse triples in reverse relations *has_part* and *part_of*. There are also 3 self-reciprocal (i.e., symmetric) relations: *verb_group*, *similar_to*, *derivationally_related _form*. 4,658 out of the 5,000 test triples have their reverse triples available in the training set. The training set itself contains 130,791 (about 92.5%) triples that are reverse of each other. On WN18, we can achieve an `FHits@1`$^{\uparrow}$of 96.4% by the aforementioned simple rule-based model (finding the relations that have more then 80% intersections) which is better than the results obtained by the embedding models (Table 3.12 of Section 3.4).

In training a knowledge graph completion model using FB15k and WN18, we fall into a form of overfitting in that the learned models are optimized for the reverse triples which cannot be generalized to realistic settings. More precisely, this is a case of excessive data leakage—the model is trained using features that otherwise would not be available when the model needs to be applied for real prediction.

3.2.2 Other Redundant Triples

(1) **FB15k**: In addition to reverse relations, there are other types of semantically redundant relations in FB15k. While it is infeasible to manually verify such semantic redundancy, we used a simple method to automatically detect it. Given two relations r_1 and r_2, we calculate how much their subject-object pairs overlap. Suppose $|r|$ is the number of instance triples in relation r and T_r denotes the set of subject-object pairs in r, i.e., $T_r =$

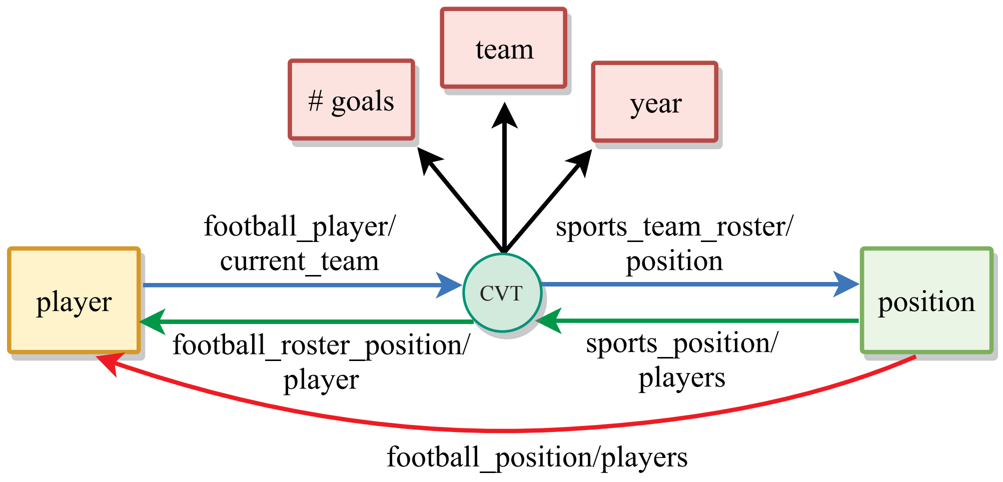


Figure 3.2: Duplicate relations

$\{(\mathsf{h},\mathsf{t}) \mid r(\mathsf{h},\mathsf{t}) \in \mathcal{G}\}$. We say r_1 and r_2 are *near-duplicate relations*, simplified as *duplicate relations*, if they satisfy the following condition:

$$\frac{|T_{r_1} \cap T_{r_2}|}{|r_1|} > \theta_1 \quad \text{and} \quad \frac{|T_{r_1} \cap T_{r_2}|}{|r_2|} > \theta_2 \tag{3.1}$$

Moreover, T_r^{-1} denotes the reverse entity pairs of T_r, i.e., $T_r^{-1} = \{(\mathsf{t},\mathsf{h}) \mid (\mathsf{h},\mathsf{t}) \in T_r\}$. We say r_1 and r_2 are *reverse duplicate relations* if they satisfy the following condition:

$$\frac{|T_{r_1} \cap T_{r_2}^{-1}|}{|r_1|} > \theta_1 \quad \text{and} \quad \frac{|T_{r_1} \cap T_{r_2}^{-1}|}{|r_2|} > \theta_2 \tag{3.2}$$

We have set θ_1 and θ_2 to 0.8 on FB15k.

For example *football_position/players* (r_1) and *sports_position/players.football_roster_position/player* (r_2) are duplicate based on this definition, since $\frac{|T_{r_1} \cap T_{r_2}|}{|r_1|} = 0.87$ and $\frac{|T_{r_1} \cap T_{r_2}|}{|r_2|} = 0.97$. These two relations are displayed in Figure 3.2 using red and green edges, respectively. The first relation records each football player's position considering their overall career. For the second relation, each instance triple is a concatenation of two edges connected through a mediator node, representing a multiary relationship about the position a player plays for a team as shown in Figure 3.2. Since most players play at the same position throughout their careers, these two relations are redundant. Another example of similar nature is that r_1 and *football_player/current_team . sports_team_roster/position* (r_3) are reverse duplicate relations, because $\frac{|T_{r_1} \cap T_{r_3}^{-1}|}{|r_1|} = 0.88$ and $\frac{|T_{r_1} \cap T_{r_3}^{-1}|}{|r_3|} = 0.97$. In Figure 3.2 they are highlighted in red and blue, respectively.

For each test triple in FB15k, we use the methods explained to determine whether it has 1) reverse triples, 2) duplicate or reverse duplicate triples in the training set, and whether it has 3) reverse triples, 4) duplicate or reverse duplicate triples in the test set itself. A triple may have redundant triples in any of these four categories. We use bitmap encoding to represent different cases of redundancy. For example, 1100 is for a triple that has both reverse triples and (reverse) duplicate triples in the training set. Hence, there are 16 possible different combinatorial cases. Considering the test triples in FB15k, not all 16 cases exist. Instead, 12 different cases exist. Figure 3.3 shows the percentages of triples in different cases. The 7 cases smaller than 1% are combined in one slice. The largest three slices are 1000 (triples with only reverse triples in the training set), 0000 (triples without any redundant triples), and 0010 (triples with only reverse triples in the test set). In total, 41,529, 1,847 and 2,701 test triples have reverse, reverse duplicate and duplicate triples in the training set, and 4,992, 249, and 328 test triples have these categories of redundant triples in the test set itself. The data redundancy causes overestimation of embedding models' accuracy. For instance, the `FMR`$^\downarrow$, `FHits@10`$^\uparrow$, `FHits@1`$^\uparrow$, and `FMRR`$^\uparrow$ of ConvE are 33.5, 88.8, 64.3, and 0.734 on such relations, whereas its performance using these measures on relations without any redundancy is only 149, 61.2, 37.3, and 0.454, respectively.

(2) **YAGO3-10**: With 1,079,040 training triples, this dataset is larger than FB15k and WN18. However, among its 37 relations, the two most populated relations *isAffiliatedTo* (r_1) and *playsFor* (r_2) account for 35% and 30% of the training triples, respectively. Although

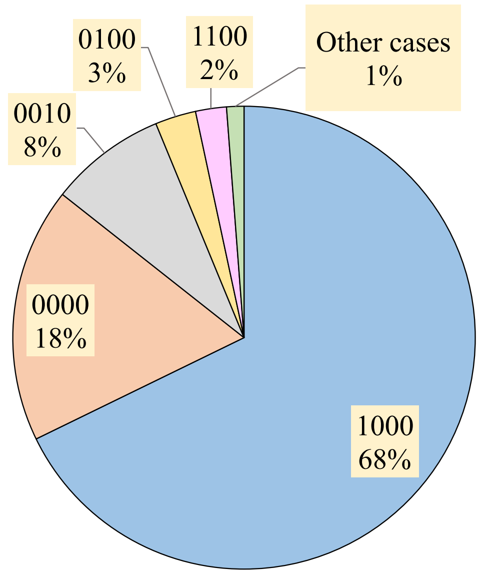


Figure 3.3: Redundancy in the test set of FB15k

r_1 semantically subsumes r_2 in the real world, they appear as near-duplicate relations in this particular dataset, as $\frac{|T_{r_1} \cap T_{r_2}|}{|r_1|} = 0.75$ and $\frac{|T_{r_1} \cap T_{r_2}|}{|r_2|} = 0.87$. In the training set, 557,696 triples find their duplicates in itself. 2,561 out of the 5,000 test triples have their duplicate triples available in the training set. The various models achieved much stronger results on r_1 and r_2 than other relations. For example, the `FMR`$^{\downarrow}$, `FHits@10`$^{\uparrow}$, `FHits@1`$^{\uparrow}$, and `FMRR`$^{\uparrow}$ of RotatE on these two relations are 225.84, 81.04, 50.43, and 0.612, in comparison with 4,540.65, 43.76, 23.38, and 0.304 on other relations. Furthermore, YAGO3-10 also has 3 semantically symmetric relations: *hasNeighbor*, *isConnectedTo*, and *isMarriedTo*. In its test set, 118 triples belonging to these relations have their reverse triples available in the training set. The `FHits@1`$^{\uparrow}$of the simple rule-based model mentioned in Section 1 is 51.6% on YAGO3-10, which outperforms embedding models, as can be seen in Table 3.11.

Table 3.1: The strong FMRR$^{\uparrow}$results on a few Cartesian product relations in FB15k-237

relation	**# of triples**	**TransE**	**DistMult**	**ComplEx**	**ConvE**	**RotatE**
/medals_awarded./medal	16	1	1	1	1	1
nutrients./nutrient	105	0.83	0.73	0.72	0.82	0.79
climate./month	60	0.98	0.77	0.95	0.98	0.98

3.3 Cartesian Product Relations

We also discovered another issue with FB15k which makes existing performance measures of embedding models unrealistic. This problem manifests in what we call *Cartesian product relations*. Given such a relation, the subject-object pairs from all instance triples of the relation form a Cartesian product. In other words, there are a set of subjects and a set of objects, and the relation exists from every subject in the first set to every object in the second set. One example Cartesian product relation is *climate*, since (a, *climate*, b) is a valid triple for every possible city a and month b. Another example is *position*, since every team in a certain professional sports league has the same set of positions. The link prediction problem for such relations thus becomes predicting whether a city has a climate in, say, January, or whether an NFL team has the quarter-back position. Such a prediction task is not very meaningful.

A few notes can be made about Cartesian product relations. (1) Similar to reverse relations and other forms of data redundancy, the existence of these relations unrealistically inflates a model's link prediction accuracy. When a substantial subset of the aforementioned subject-object Cartesian product is available in the training set, it is relatively easy for a model to attain strong accuracy. However, it is problematic to mix such straightforward test cases with more realistic, challenging cases. At least, the performance of a model

Table 3.2: Link prediction using Cartesian product property

	TransE results							
	FB15k as ground truth							
	MR$^{\downarrow}$	H10$^{\uparrow}$	H1$^{\uparrow}$	MRR$^{\uparrow}$	FMR$^{\downarrow}$	FH10$^{\uparrow}$	FH1$^{\uparrow}$	FMRR$^{\uparrow}$
r1	19.97	43.33	3.33	0.14	1.39	99.17	83.33	0.9
r2	5	100	0	0.24	2.5	100	0	0.42
r3	6.5	100	0	0.22	2.5	100	0	0.42
r4	1784.25	25	0	0.09	1777	25	0	0.1
r5	12.74	58.82	11.76	0.31	1	100	100	1
r6	10.72	62.5	9.38	0.27	1.03	100	96.88	0.98
r7	7.75	50	25	0.35	2.75	100	25	0.51
r8	10.75	62.5	0	0.19	4.63	75	25	0.44
r9	15.59	53.13	18.75	0.34	1	100	100	1

	Prediction using Cartesian product property											
	FB15k as ground truth								**Freebase as ground truth**			
	MR$^{\downarrow}$	H10$^{\uparrow}$	H1$^{\uparrow}$	MRR$^{\uparrow}$	FMR$^{\downarrow}$	FH10$^{\uparrow}$	FH1$^{\uparrow}$	FMRR$^{\uparrow}$	FMR$^{\downarrow}$	FH10$^{\uparrow}$	FH1$^{\uparrow}$	FMRR$^{\uparrow}$
r1	16.52	52	6	0.18	1.46	100	70	0.83	1.40	100	71	0.84
r2	3.6	100	35	0.55	1.6	100	70	0.80	1.45	100	75	0.84
r3	3.6	100	40	0.57	1.45	100	70	0.82	1.4	100	70	0.83
r4	995.25	74	25	0.45	990.17	88	88	0.88	990.09	88	88	0.88
r5	364.03	60	17	0.35	352.03	94	94	0.94	352.03	94	94	0.94
r6	9.62	64	12	0.31	1	100	100	1	1	100	100	1
r7	1872.38	60	10	0.3	1865.72	75	25	0.43	1864.83	75	53	0.6
r8	6.63	74	14	0.37	2.69	100	60	0.7	2.09	100	64	0.74
r9	11.57	65	19	0.36	1	100	100	1	1	100	100	1

Table 3.3: Cartesian product relations used in Table 3.2

r1	*travel_destination/climate . travel_destination_monthly_climate/month*
r2	*computer_videogame/gameplay_modes*
r3	*gameplay_mode/games_with_this_mode*
r4	*educational_institution/sexes_accepted . gender_enrollment/sex*
r5	*olympic_medal/medal_winners . olympic_medal_honor/olympics*
r6	*x2010fifaworldcupsouthafrica/world_cup_squad/current_world_cup_squad . x2010fifaworldcupsouthafrica/current_world_cup_squad/position*
r7	*dietary_restriction/compatible_ingredients*
r8	*ingredient/compatible_with_dietary_restrictions*
r9	*olympic_games/medals_awarded . olympic_medal_honor/medal*

should be separately evaluated on Cartesian product relations and non-Cartesian product relations.

(2) Albeit not always meaningful, one may still perform link prediction on Cartesian product relations. However, a simpler approach can be more effective than learning complex embedding models. For instance, by examining all instance triples of a relation r, a method can identify the sets of all subjects $S_r = \{\mathtt{h} \mid \exists r(\mathtt{h}, \mathtt{t}) \in \mathcal{G}\}$ and objects $O_r = \{\mathtt{t} \mid \exists r(\mathtt{h}, \mathtt{t}) \in \mathcal{G}\}$ in the instance triples. By observing a large percentage of possible subject-object pairs existing in the relation, the method can derive the relation might be a Cartesian product relation. More specifically, if $|r|\ /\ (|S_r| \times |O_r|)$ is greater than a pre-determined threshold (0.8 in our study), we consider r a Cartesian product relation. There are a total of 31,771 relations (375,387,927 instance triples) in the aforementioned May 2013 Freebase snapshot, of which 3,568 relations have only one instance triple each. Among the 28,203 remaining relations (375,384,359 instance triples), we detected 2,951 Cartesian product relations (2,605,338 instance triples) using this method. We also iden-

tified 142 Cartesian product relations in FB15k, with 13,038 triples. Although there are not as many Cartesian product relations as reverse relations, we discovered that among the relations on which embedding models attained the highest accuracy there are Cartesian product relations. Table 3.1 shows such results on the relations using $\texttt{FMRR}^{\uparrow}$ on FB15k after reverse triples are removed. [5] These are the Cartesian product relations among the top-12 relations ranked by $\texttt{FMRR}^{\uparrow}$ of ConvE on FB15k-237.

Once a relation r is detected as a Cartesian product relation, for link prediction, we can predict triple (h, r, t) to be valid, given any $\mathsf{h} \in S_r$ and $\mathsf{t} \in O_r$. We can further extend this approach. If an entity type system exists (which is the case with Freebase), we can identify the common type of all entities in S_r and O_r, respectively, and then predict (h, r, t) valid for all h and t belonging to the corresponding types.

(3) The existence of Cartesian product relations in FB15k is quite artificial. In fact, many such relations are due to mediator nodes in Freebase and simplification in FB15k for removing such nodes (see Section 3.1). The majority of relationships in Freebase are multiary relationships connected through mediator nodes. For instance, a mediator node is connected to Tokyo with an edge labeled *climate* and to January with an edge labeled *month*. It is further connected to 34 with an edge labeled *average_min_temp_c*, indicating that the average low temperature in Tokyo is 34 degrees Fahrenheit in January. In fact, it is also connected to other nodes for capturing the maximal temperature, rain fall, and so on. The more realistic and useful prediction task is to predict the average temperature, rather than whether a city has a temperature. Even though most real-world relationships are multiary, the studies on link prediction have often simplified it as multiple binary relationships

[5]This dataset is called FB15k-237 and will be further discussed in Section 3.4. We want to inspect the impact of Cartesian product relations after removing reverse relations, because the latter also causes overestimation of embedding models' accuracy and they dominate Cartesian product relations in terms of number of triples.

(which is lossful as the multiple binary relationships cannot be used to restore the identical original relationship). That is how FB15k entails the less meaningful prediction tasks. In fact, out of the 2,951 Cartesian product relations mentioned in (2), 1,766 are concatenated relations. In the specific example above, the concatenated edge is between Tokyo and January, connecting the original *climate* and *month* edges.

(4) The performance measures in Section 2.3.2 are based on the closed-world assumption and thus have flaws when a model correctly predicts a triple that does not exist in the ground-truth dataset. More specifically, if a corrupted triple of a given test triple does not exist in the training, test, or validation set, it is considered incorrect. However, the corrupted triple might be correct as well. If a model ranks the corrupted triple higher than the test triple itself, its accuracy measures will be penalized, which contradicts with the exact goal of link prediction—finding correct triples that do not already exist in the dataset. While this defect of the accuracy measures is applicable on all types of relation, [6] it is more apparent in evaluating a method that is capable of leveraging the characteristics of Cartesian product relations. Such a method would mark many triples non-existent in the dataset as correct and further rank many of them higher than the test triples.

Table 3.2 uses several measures to show the accuracy of the prediction method based on the Cartesian product property, as explained in (2). [7] The method's accuracy is evaluated using both FB15k and the larger Freebase snapshot as the ground truth. The table also shows the results of using TransE. In all cases FB15k training set is used as the training data for making predictions. The table presents the results on all 9 Cartesian product relations we detected from FB15k, which are listed in Table 5.9. Some of them are detected as Cartesian product relations by applying the aforementioned process over the training set of

[6] Fundamentally it is because the models are evaluated by ranking instead of binary classification of triples.

[7] For coping with space limitations, we shortened the names of some measures, e.g., `FHits@10`$^\uparrow$is shortened as `FH10`$^\uparrow$.

FB15k and some are detected over the Freebase snapshot. We can make several observations regarding the results in Table 3.2. First, the performance of using Cartesian product property is higher when Freebase instead of FB15k is the ground truth. This is because Freebase subsumes FB15k and thus is affected less by the defect mentioned in (4). Second, using Cartesian product property is more accurate than embedding models such as TransE, especially when the Freebase snapshot is used as the ground truth to calculate filtered measures. (Note that using Freebase as the ground truth will not affect unfiltered measures such as $\texttt{MR}^{\downarrow}$. Therefore we do not repeat those measures in the table.) For example, consider predicting triples in relation *r2*. We observed that the Cartesian product property attained a $\texttt{FMRR}^{\uparrow}$ of 0.80 using FB15k as ground truth, in comparison to 0.42 by TransE. The accuracy is further improved to 1 when using the Freebase snapshot as the ground truth.

3.4 Experiments

3.4.1 FB15k-237, WN18RR, YAGO3-10-DR

Among the first to document the data redundancy in FB15k, Toutanova and Chen [21] created FB15k-237 from FB15k by removing such redundancy. They first limited the set of relations in FB15k to the most frequent 401 relations. Their approach of removing redundancy is essentially the same as the equations for detecting duplicate and reverse duplicate relations in Section 3.2.2, likely with different thresholds. For each pair of such redundant relations, only one was kept. This process decreased the number of relations to 237. They also removed all triples in test and validation sets whose entity pairs were directly linked in the training set through any relation. This step could incorrectly remove useful information. For example, *place_of_birth* and *place_of_death* may have many overlapping subject-object pairs, but they are not semantically redundant. Furthermore, the creation of FB15k-237 did not resort to the absolutely accurate reverse relation information encoded by *reverse_property*. Finally, it does not identify Cartesian product relations. Nevertheless, we used both FB15k-237 and FB15k in our experiments. This allows us to corroborate the experiment results with those from a number of recent studies.

To remove the WN18 reverse relations mentioned in Section 3.2, Dettmers et al. [22] created WN18RR by keeping just one relation from each pair of reverse relations. The resulting dataset WN18RR has 40,943 entities in 11 relations. However, this dataset still contains symmetric (i.e., self-reciprocal) relations—a special case of reverse relations where a relation is the reverse of itself. Particularly, more than 34% of the training triples in WN18RR belong to the symmetric relation *derivationally_related_form* which is for terms in different syntactic categories that have the same morphological root. For instance, both triples (question, *derivationally_related_form*, inquire) and (inquire, *derivationally_related_form*, question) are in the training set. Among the 11 relations in WN18RR's training set, 3 are self-reciprocal, which account for 30,933 of the 86,835 training triples. 28,835 out of these 30,933 triples form reverse pairs. For the remaining 2,098 triples, 1,052 form reverse pairs of with 1,052 triples (around 33.57%) of the test set.

As mentioned in Section 3.2, YAGO3-10 has two near-duplicate relations *isAffiliatedTo* and *playsFor* and 3 semantically symmetric relations *hasNeighbor*, *isConnectedTo*, and *isMarriedTo*. We removed *playsFor*. In the training set, for each pair of redundant triples belonging to symmetric relations, we only kept one. Moreover, we removed a triple from the test and validation sets if it belongs to any symmetric relation and its entity pairs are directly linked in the training set. We call the resulting dataset YAGO3-10-DR, of which the statistics can be found in Table 2.2.

3.4.2 Experiment Setup

Our experiments were conducted on an Intel-based machine with an Intel Xeon E5-2695 processor running at 2.1GHz, Nvidia Geforce GTX1080Ti GPU, and 256 GB RAM.

The experiments used source codes of various methods from several places, including the OpenKE [44] repository which covers implementations of TransE, TransH, TransR, TransD, RESCAL, DistMult, and ComplEx, as well as the source code releases of ComplEx (which also covers DistMult), ConvE, RotatE, and TuckER. The URLs of these implementations can be found in Table 3.4.

The different models used in our experiments have different hyperparameters. We used the same hyperparameter settings for FB15k, FB15k-237, WN18, WN18RR, and YAGO3-10 that were used by the developers of the source codes. The details can be found in the codes.

We also experimented with rule-based system AMIE [18]. AMIE rules were generated by applying the AMIE+ (`https://bit.ly/2Vq2OIB`) code released by the authors of [45] on the training sets of FB15k, FB15k-237, WN18, WN18RR, and YAGO3-10. The parameters of AMIE+ were set in the same way as in [46], for all datasets. For any link prediction task (h, *r*, ?) or (?, *r*, t), all the rules that have relation *r* in the rule head are employed. The instantiations of these rules are used to generate the ranked list of results. For example, for test case (Bill Gates, *place_of_birth*, ?), the following rule will be employed: (?a, *places_lived/location*, ?b) $\Rightarrow$ (?a, *place_of_birth*, ?b). Then the instantiations of variable ?b are used to find the list of predictions. Several rules may generate the same answer entity. It is imperative to combine the confidence of those rules in some way in order to score the answer entities. We ranked the answer entities by the maximum confidence of the rules instantiating them and broke ties by the number of applicable rules [46].

3.4.3 Results

Tables 3.4 and 3.5 display the results of link prediction on FB15k vs. FB15k-237 and WN18 vs. WN18RR for all compared methods, using both raw and filtered metrics explained in Section 2.3.2. For each method, the table shows the original publication where

Table 3.4: Link prediction results on FB15k and FB15k-237

	FB15k						**FB15k-237**					
	Raw measures			**Filtered measures**			**Raw measures**			**Filtered measures**		
Model	MR$^\downarrow$	H10$^\uparrow$	MRR$^\uparrow$	FMR$^\downarrow$	FH10$^\uparrow$	FMRR$^\uparrow$	MR$^\downarrow$	H10$^\uparrow$	MRR$^\uparrow$	FMR$^\downarrow$	FH10$^\uparrow$	FMRR$^\uparrow$
TransE [14]	243.0 199.9	34.9 44.3	– 0.227	125.0 68.8	47.1 62.4	– 0.391	– 363.3	– 32.2	– 0.169	– 223.4	– 47.5	– 0.288
TransH [31]	211.0 234.7	42.5 45.5	– 0.177	84.0 84.0	58.5 69.0	– 0.346	– 398.8	– 30.9	– 0.157	– 251.1	– 49.0	– 0.290
TransR [32]	226.0 231.9	43.8 48.8	– 0.236	78.0 78.2	65.5 72.9	– 0.471	– 391.3	– 31.4	– 0.164	– 240.2	– 51.0	– 0.314
TransD [33]	211.0 234.4	49.4 47.4	– 0.179	67.0 85.4	74.2 70.9	– 0.352	– 391.6	– 30.6	– 0.154	– 244.1	– 48.5	– 0.284
DistMult [34]	– 313.0 264.3	– 45.1 50.3	– 0.206 0.240	– 159.6 106.3	57.7 71.4 82.8	0.35 0.423 0.651	– 566.3 –	– 30.3 –	– 0.151 –	– 418.5 –	– 41.8 –	– 0.238 –
ComplEx [35]	– 350.3 250.6	– 43.8 49.2	0.242 0.205 0.233	– 192.3 90.0	84.0 72.7 83.2	0.692 0.516 0.685	– 656.4 –	– 29.9 –	– 0.158 –	– 508.5 –	– 42.3 –	– 0.249 –
ConvE [22]	– 189.5	– 51.7	– 0.268	64.0 46.5	87.3 85.6	0.745 0.698	– 481.7	– 28.6	– 0.154	246.0 271.5	49.1 48.1	0.316 0.305
RotatE [26]	– 190.4	– 50.6	– 0.256	40 41.1	88.4 88.1	0.797 0.791	– 333.4	– 31.7	– 0.169	177 179.1	53.3 53.2	0.338 0.337
TuckER [25]	– 186.4	– 51.3	– 0.260	– 39.0	89.2 89.1	0.795 0.790	– 343.5	– 35.4	– 0.197	– 164.8	54.4 53.9	0.358 0.355
AMIE [18]	337.0	64.6	0.370	309.7	88.1	0.797	1909	36.2	0.201	1872	47.7	0.308

● Published results ● OpenKE (`https://github.com/thunlp/OpenKE`) ● ComplEx (`https://github.com/ttrouill/complex`) ● ConvE (`https://github.com/TimDettmers/ConvE`) ● RotatE (`https://github.com/DeepGraphLearning/KnowledgeGraphEmbedding`) ● TuckER (`https://github.com/ibalazevic/TuckER`) ● AMIE (produced by us)

Table 3.5: Link prediction results on WN18 and WN18RR

	WN18						WN18RR					
	Raw measures			Filtered measures			Raw measures			Filtered measures		
Model	MR$^{\downarrow}$	H10$^{\uparrow}$	MRR$^{\uparrow}$	FMR$^{\downarrow}$	FH10$^{\uparrow}$	FMRR$^{\uparrow}$	MR$^{\downarrow}$	H10$^{\uparrow}$	MRR$^{\uparrow}$	FMR$^{\downarrow}$	FH10$^{\uparrow}$	FMRR$^{\uparrow}$
TransE [14]	263.0 142.4	75.4 75.4	– 0.395	251.0 130.8	89.2 86.0	– 0.521	– 2414.7	– 47.2	– 0.176	– 2401.3	– 51.0	– 0.224
TransH [31]	318.0 190.1	75.4 76.2	– 0.434	303.0 178.7	86.7 86.1	– 0.570	– 2616	– 46.9	– 0.178	– 2602	– 50.4	– 0.224
TransR [32]	232.0 199.7	78.3 77.8	– 0.441	219.0 187.9	91.7 87.3	– 0.583	– 2847	– 48.1	– 0.184	– 2834	– 51.0	– 0.235
TransD [33]	242.0 202.5	79.2 79.5	– 0.421	229 190.6	92.5 91.0	– 0.569	– 2967	– 47.4	– 0.172	– 2954	– 50.6	– 0.219
DistMult [34]	– 452.9 915.0	– 80.9 80.7	– 0.531 0.558	– 438.5 902.1	94.2 93.9 93.5	0.83 0.759 0.834	– 3798.1 –	– 46.2 –	– 0.264 –	– 3784 –	– 47.5 –	– 0.371 –
ComplEx [35]	– 477.3 636.1	– 81.5 80.5	0.587 0.597 0.584	– 462.7 622.7	94.7 94.6 94.5	0.941 0.902 0.940	– 3755.9 –	– 46.7 –	– 0.276 –	– 3741.7 –	– 47.4 –	– 0.398 –
ConvE [22]	– 413.1	– 80.6	– 0.574	504 396.6	95.5 95.5	0.94 0.945	– 5007.3	– 47.9	– 0.261	5277 4992.7	48.0 50.4	0.46 0.429
RotatE [26]	– 286.2	– 81.1	– 0.584	309 269.7	95.9 96.0	0.949 0.950	– 3374	– 53.0	– 0.306	3340 3359.8	57.1 57.3	0.476 0.476
TuckER [25]	– 484.7	– 80.6	– 0.576	– 468.1	95.8 95.8	0.953 0.950	– 6598	– 46.8	– 0.272	– 6584	52.6 50.2	0.470 0.451
AMIE [18]	1299.8	94.0	0.931	1299.1	94.0	0.940	12963	35.6	0.357	12957	35.6	0.357

● Published results ● OpenKE (`https://github.com/thunlp/OpenKE`) ● ComplEx (`https://github.com/ttrouill/complex`) ● ConvE (`https://github.com/TimDettmers/ConvE`) ● RotatE (`https://github.com/DeepGraphLearning/KnowledgeGraphEmbedding`) ● TuckER (`https://github.com/ibalazevic/TuckER`) ● AMIE (produced by us) it comes from. The values in black color are the results listed in the original publication, while a hyphen under a measure indicates that the original publication did not list the corresponding value. The values in other colors are obtained through our experiments using various source codes, as listed in the tables.

Below we summarize and explain the results in Table 3.4 and Table 3.5. (1) The overall observation is that the performance of all methods worsens considerably after removal of reverse relations. For instance, the $\texttt{FMRR}^{\uparrow}$ of ConvE—one of the best performing methods under many of the metrics—has decreased from 0.698 (on FB15k) to 0.305 (on FB15k-237) and from 0.945 (on WN18) to 0.429 (on WN18RR). Its $\texttt{FMR}^{\downarrow}$ also became much worse, from 46.5 (FB15k) to 271.5 (FB15k-237) and from 396.6 (WN18) to 4992.7 (WN18RR). This result verifies that embedding-based methods may only perform well on reverse relations. However, a straightforward approach based on detection of reverse relations can achieve comparable or even better accuracy, as explained in Section 3.2.1.

(2) Many successors of TransE (e.g., DistMult, ComplEx, ConvE, RotatE, and TuckER) were supposed to significantly outperform it. This was indeed verified by our experiment results on FB15k and WN18. However, on FB15k-237, their margin over TransE became much smaller. For example, by $\texttt{FMRR}^{\uparrow}$, TransE's accuracy is 0.288, in comparison with DistMult (0.238), ComplEx (0.249), ConvE (0.305), RotatE (0.337), and TuckER (0.355). We hypothesize that these models improved the results mostly on reverse and duplicate triples and hence, after removing those triples, they do not exhibit clear advantage. This hypothesis can be verified by our finding that most of the test triples on which these models outperformed TransE have reverse or duplicate triples in the training set, as shown in Table 3.6. The observations regarding WN18RR are similar, although the successors of TransE demonstrated wider edge over TransE. We note that, however, this could be attributed to the large number of reverse triples from symmetric relations that are retained in WN18RR, as explained in Section 3.4.1.

Table 3.6: Percentages of test triples, among those on which various models outperformed TransE, that have reverse and duplicate triples in training set

FB15k				
Model	`FMR`$^{\downarrow}$	`FHits@10`$^{\uparrow}$	`FHits@1`$^{\uparrow}$	`FMRR`$^{\uparrow}$
DistMult	82.17 %	90.78 %	95.16 %	85.88 %
ComplEx	81.24 %	90.14 %	94.98 %	84.67 %
ConvE	78.69 %	87.12 %	91.1 %	78.04 %
RotatE	78.61%	88.37%	94.41%	78.16%
TuckER	78.96%	87.76%	93.65%	78.87%
WN18				
Model	`FMR`$^{\downarrow}$	`FHits@10`$^{\uparrow}$	`FHits@1`$^{\uparrow}$	`FMRR`$^{\uparrow}$
DistMult	98.43 %	99.4 %	98.53 %	98.55 %
ComplEx	97.93 %	98.72 %	99.1 %	97.73 %
ConvE	96.42 %	96.7 %	98.96 %	95.85 %
RotatE	95.17%	95.82%	98.74%	94.00%
TuckER	95.75%	95.18%	98.45%	95.70%

(3) Tables 3.4 and 3.5 show that the performance of AMIE also substantially degenerates in the absence of data redundancy. For example, its `FHits@10`$^{\uparrow}$ has decreased from 88.1% (FB15k) to 47.7% (FB15k-237) and from 94.0% (WN18) to 35.6% (WN18RR).

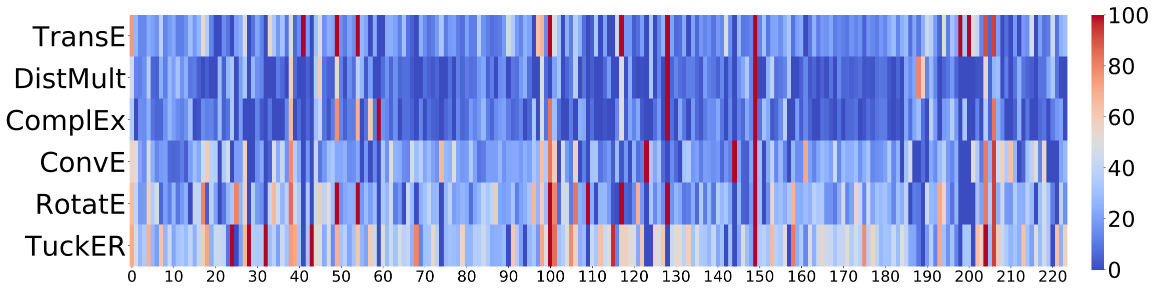
(4) We further analyzed how the most-accurate models perform. There are 224, 11, and 34 distinct relations in the test sets of FB15k-237, WN18RR, and YAGO3-10, respectively. Table 3.7 shows, for each metric and each model, the number of distinct test relations

Table 3.7: Number of relations on which each model is the most accurate

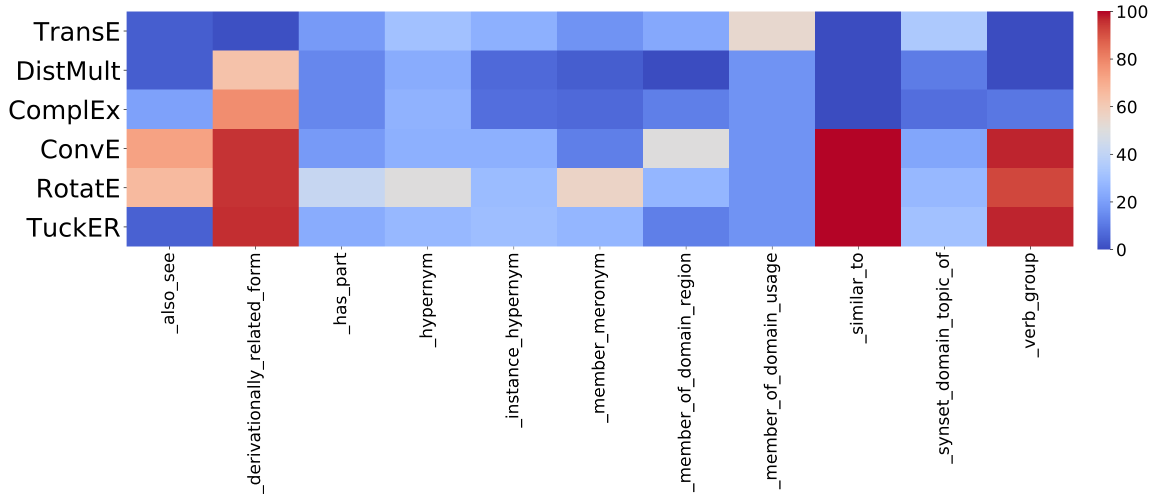
	FB15-237				WN18RR				YAGO3-10			
Model	FMR$^{\downarrow}$	FH10$^{\uparrow}$	FH1$^{\uparrow}$	FMRR$^{\uparrow}$	FMR$^{\downarrow}$	FH10$^{\uparrow}$	FH1$^{\uparrow}$	FMRR$^{\uparrow}$	FMR$^{\downarrow}$	FH10$^{\uparrow}$	FH1$^{\uparrow}$	FMRR$^{\uparrow}$
TransE	17	43	23	19	8	2	0	1	9	10	3	3
DistMult	5	23	13	6	0	1	0	0	4	3	6	1
ComplEx	6	20	12	5	0	2	0	0	6	10	7	4
ConvE	8	41	30	15	1	3	3	3	4	7	7	2
RotatE	77	92	58	55	2	9	4	5	2	13	8	5
TuckER	95	112	101	91	2	2	6	4	5	13	10	10
AMIE	31	66	68	49	1	2	3	3	4	6	12	9

on which the model is the most accurate.[8] Furthermore, the heatmap in Figure 3.4a (Figures 3.4b, and 3.4c resp.) shows, for each of the 224 (11, 34 resp.) relations in FB15k-237 (WN18RR, YAGO3-10 resp.), the percentage of test triples on which each model has the best performance (i.e., the highest rank) in comparison with other models, using FMRR$^{\uparrow}$ as the performance measure. What is particularly insightful about Figure 3.4b is that TuckER, RotatE, and ConvE clearly dominated other models on relations *derivationally_related_form*, *similar_to*, and *verb_group*. As explained in Section 3.4.1, these are all symmetric relations where reverse triples are retained in WN18RR. This observation corroborates with the analysis in 2) above. It suggests that the state-of-the-art models might be particularly optimized for reverse triples. On the other hand, the simple rule based on data statistics can attain an FHits@1$^{\uparrow}$ of 97.85% on these 3 relations.

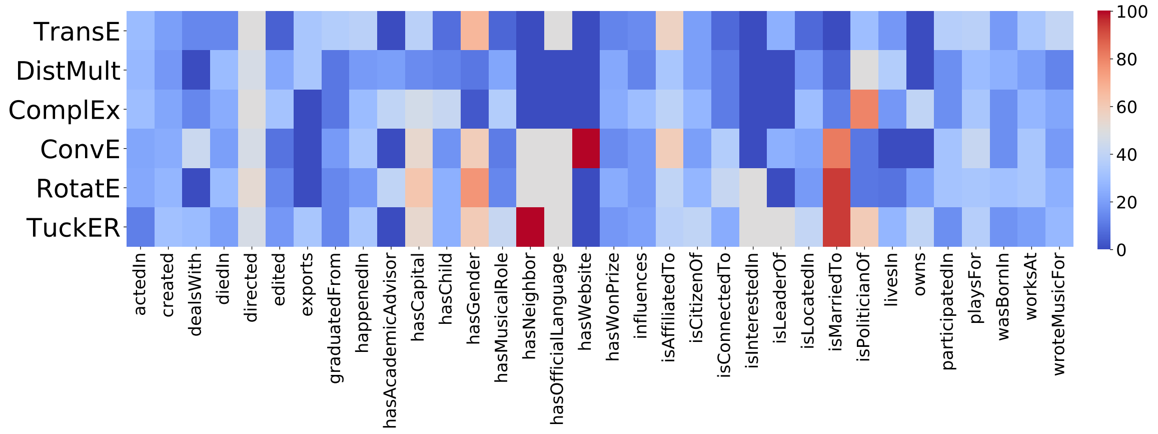
[8]We rounded all accuracy measures to the nearest hundredth except for MRR/FMRR which are rounded to the nearest thousandth. Since there are ties in best performing models, the summation of each column can be greater than 224, 11, and 34.



(a) FB15k-237 results



(b) WN18RR results

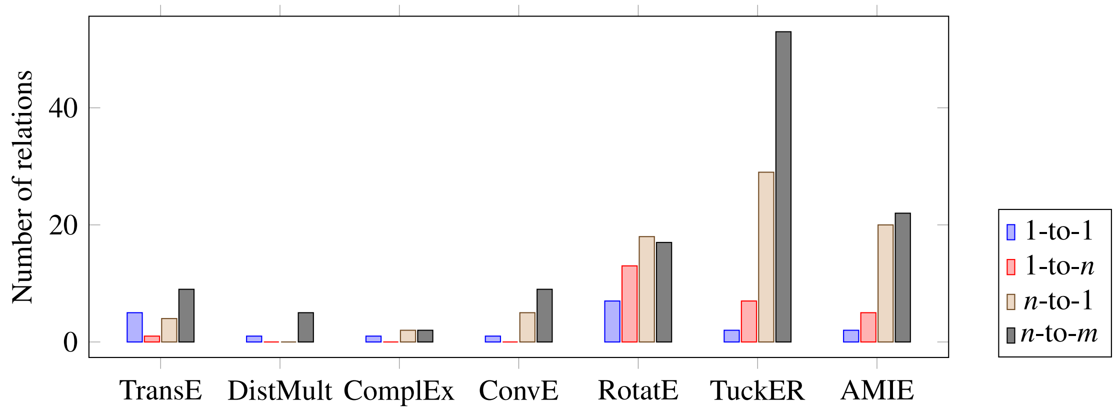


(c) YAGO3-10 results

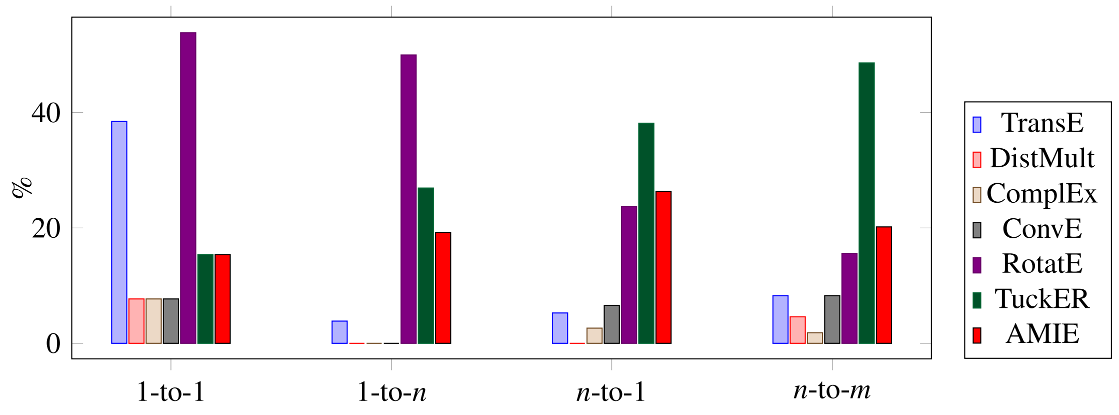
Figure 3.4: Percentage of triples on which each method outperforms others, for each relation

(5) To better understand the strengths and weaknesses of each model, we further broke down the numbers in the column `FMRR`$^\uparrow$of Table 3.7. The relations are categorized into 4 different classes: 1-to-1, 1-to-*n*, *n*-to-1 and *n*-to-*m*, based on the average number of heads per tail and tails per head. An average number less than 1.5 is marked as "1" and "*n*" otherwise [14]. Among the 224 distinct relations in the test set of FB15k-237, 5.8% are 1-to-1, 11.6% are 1-to-*n*, 33.9% are *n*-1, and 48.7% are *n*-to-*m* relations. The numbers of test triples belonging to these 4 types of relations are 192, 1,293, 4,285 and 14,696, respectively. In WN18RR, the 11 distinct relations in the test set are distributed as 2, 4, 3, and 2 in these four classes, and the numbers of test triples are 42, 475, 1,487, and 1,130, respectively. Figure 3.5a shows the break-down of relations on which each method has the best result for FB15k-237. Figure 3.5b shows the break-down of best performing models within each type of relations. Overall, RotatE and TuckER outperformed other models, with RotatE particularly excelling on 1-to-1 and 1-to-*n* relations and TuckER on *n*-to-1 and *n*-to-*m* relations. TransE still demonstrated its robust strength on 1-to-1 relations. Figure 3.6 shows the results on WN18RR. We can see that TransE has good performance on 1-to-*n* relations, and while Tucker is one of the best performing models on this dataset, it is weaker than other models on 1-to-*n* relations.

(6) We further computed the `FHits@10`$^\uparrow$ for head and tail predictions separately for each relation type of FB15k-237 and WN18RR, as in Tables 3.8 and 3.9. The first and second best results are shown with boldface and underline, respectively. All the methods performed better at predicting the "1" side of 1-to-*n* and *n*-to-1 relations on both datasets. On FB15k-237, RotatE and TransE are the first and second best performing models on 1-to-1 relations, respectively. RoataE and TuckER have the highest performance on 1-to-*n*, *n*-to-1, and *n*-to-*m* relations. Performance of TuckER, RotatE, and AMIE is the best in predicting the side *n* of 1-to-*n* and *n*-to-1 relations which are more complicated. On WN18RR, almost all models have very high accuracy on 1-to-1 and *n*-to-*m* relations. Note



(a) Categorizing the relations on which each method has the best result

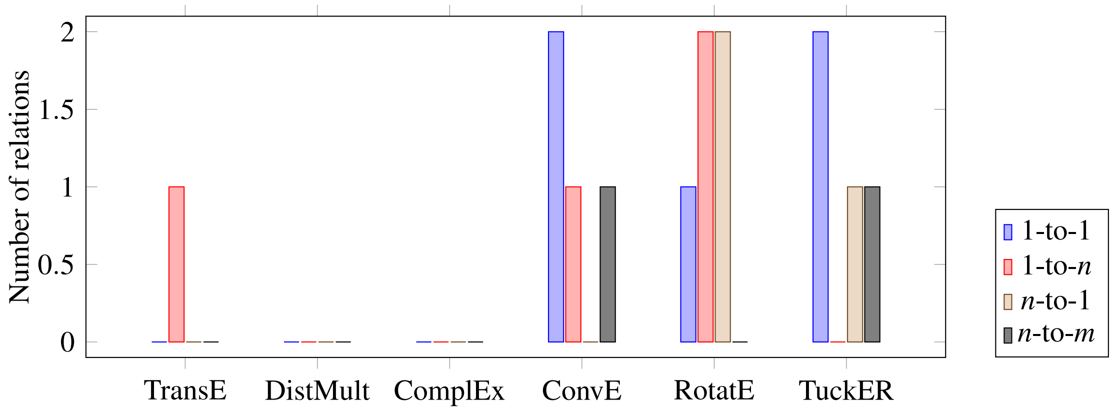


(b) Break-down of methods achieving best performance on each type of relations

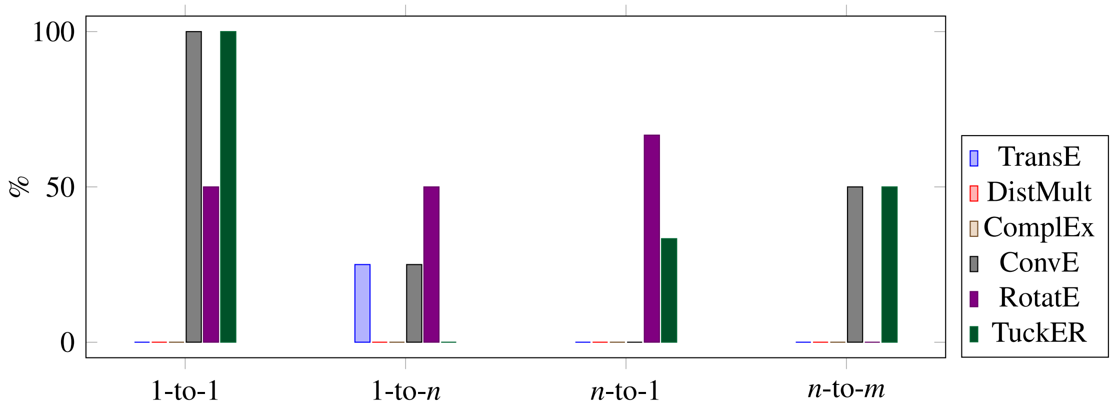
Figure 3.5: Models with best `FMRR`$^\uparrow$ on FB15k-237

that self-reciprocal relations *derivationally_related_form*, *similar_to*, and *verb_group* belong to these categories.

(7) As mentioned in Section 3.2.2, YAGO3-10 is dominated by two relations *isAffiliatedTo* and *playsFor* which are effectively duplicate relations. The results on this dataset for some the best performing models are shown in Table 3.11. AMIE achieved better performance than embedding models on `FHits@1`$^\uparrow$and `FMRR`$^\uparrow$. Similar to (5) and (6), we generated detailed results on this dataset, shown in Figure 3.7 and Table 3.10. Figure 3.7 shows that



(a) Categorizing the relations on which each method has the best result



(b) Break-down of methods achieving best performance on each type of relations

Figure 3.6: Models with best $\texttt{FMRR}^{\uparrow}$ on WN18RR

AMIE outperformed other models on 1-to-1 and *n*-to-*m* relations while TuckER was on par with AMIE on 1-to-1 relations. RotatE and TuckER outperformed others in 1-to-*n* and *n*-to-1 relations. Table 3.10 shows that embedding models have very similar results particularly on 1-to-1 and *n*-to-*m* relations. We note that the duplicate relations *isAffilatedTo* and *playsFor* belong to *n*-to-*m* and the self-reciprocal relation *isMarriedTo* belongs to 1-to-1.

(8) The results on YAGO3-10-DR are also shown in Table 3.11. Same as what we observed on FB15k-237 and WN18RR, the performance of all models dropped significantly

Table 3.8: FHits@10$^{\uparrow}$ by category of relations on FB15k-237

	1-**to**-1		1-**to**-*n*		*n*-**to**-1		*n*-**to**-*m*	
Model	**Left** FH10$^{\uparrow}$	**Right** FH10$^{\uparrow}$	**Left** FH10$^{\uparrow}$	**Right** FH10$^{\uparrow}$	**Left** FH10$^{\uparrow}$	**Right** FH10$^{\uparrow}$	**Left** FH10$^{\uparrow}$	**Right** FH10$^{\uparrow}$
TransE	<u>55.21</u>	<u>55.21</u>	60.40	9.59	11.74	84.60	40.52	56.12
DistMult	47.92	46.35	43.62	3.63	4.34	76.06	36.09	51.48
ComplEx	47.40	46.88	36.66	3.94	5.13	75.82	36.95	52.48
ConvE	27.60	26.56	59.63	11.29	14.38	85.34	41.20	56.73
RotatE	**59.38**	**58.85**	**67.52**	<u>13.61</u>	17.08	<u>87.49</u>	<u>47.38</u>	<u>61.49</u>
TuckER	<u>55.21</u>	52.60	<u>65.58</u>	**15.62**	**21.52**	**87.84**	**48.14**	**61.54**
AMIE	43.23	44.27	46.17	13.07	<u>18.20</u>	80.49	42.45	55.19

after removal of duplicate and reverse triples. Hence, we recommend YAGO3-10-DR for evaluating embedding models instead of YAGO3-10.

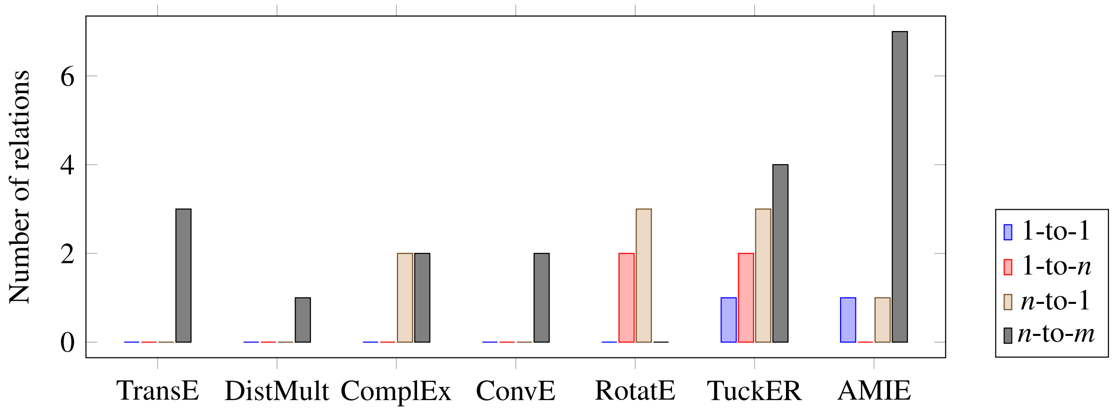
(9) Table 3.12 compares various methods using FHits@1$^{\uparrow}$, which is a more demanding measure than FHits@10$^{\uparrow}$, since it only considers whether a model ranks a correct answer at the very top. The results show that embedding models and AMIE have comparable performance on FB15k and WN18 as these datasets contain relations that clearly can be predicted by rules. On FB15k-237 and WN18RR, embedding models RotatE and TuckER stand out.

Table 3.9: `FHits@10`$^{\uparrow}$ by category of relations on WN18RR

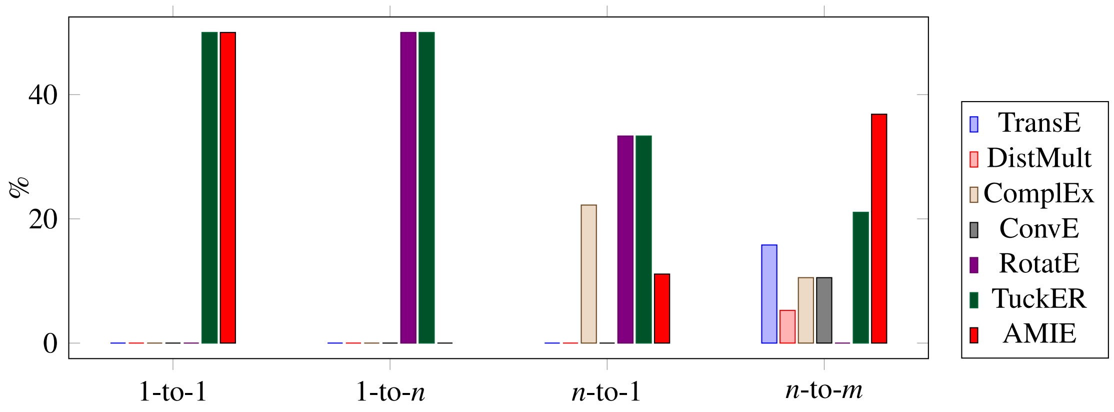
	1-to-1		**1-to-*n***		***n*-to-1**		***n*-to-*m***	
Model	**Left** `FH10`$^{\uparrow}$	**Right** `FH10`$^{\uparrow}$	**Left** `FH10`$^{\uparrow}$	**Right** `FH10`$^{\uparrow}$	**Left** `FH10`$^{\uparrow}$	**Right** `FH10`$^{\uparrow}$	**Left** `FH10`$^{\uparrow}$	**Right** `FH10`$^{\uparrow}$
TransE	<u>92.86</u>	<u>92.86</u>	42.32	15.79	14.32	<u>34.90</u>	92.92	93.72
DistMult	**97.62**	<u>92.86</u>	24.21	6.32	5.78	33.83	95.75	**95.75**
ComplEx	**97.62**	**97.62**	29.47	7.58	5.72	33.02	<u>95.84</u>	95.31
ConvE	**97.62**	**97.62**	<u>44.63</u>	16.84	11.57	31.34	95.22	94.96
RotatE	**97.62**	**97.62**	**53.68**	**28.84**	**20.98**	**43.04**	**96.11**	<u>95.58</u>
TuckER	**97.62**	**97.62**	40.21	<u>18.95</u>	<u>15.13</u>	30.26	94.96	91.77
AMIE	**97.62**	**97.62**	1.26	1.26	1.41	1.41	92.83	92.83

Table 3.10: `FHits@10`$^{\uparrow}$ by category of relations on YAGO31-0

	1-to-1		**1-to-*n***		***n*-to-1**		***n*-to-*m***	
Model	**Left** `FH10`$^{\uparrow}$	**Right** `FH10`$^{\uparrow}$	**Left** `FH10`$^{\uparrow}$	**Right** `FH10`$^{\uparrow}$	**Left** `FH10`$^{\uparrow}$	**Right** `FH10`$^{\uparrow}$	**Left** `FH10`$^{\uparrow}$	**Right** `FH10`$^{\uparrow}$
TransE	76.67	80.00	48.31	32.58	3.89	**78.34**	63.82	80.23
DistMult	83.33	<u>83.33</u>	<u>49.44</u>	29.21	6.09	55.67	60.21	79.77
ComplEx	**90.00**	<u>83.33</u>	**55.06**	31.46	5.75	29.10	62.26	80.47
ConvE	83.33	80.00	43.82	**39.33**	3.21	74.11	<u>65.31</u>	**80.98**
RotatE	83.33	<u>83.33</u>	**55.06**	<u>38.20</u>	**6.43**	<u>77.66</u>	61.66	<u>80.68</u>
TuckER	<u>86.67</u>	**90.00**	42.70	**39.33**	5.25	72.59	60.72	79.67
AMIE	73.33	73.33	13.48	11.24	<u>6.26</u>	7.45	**67.44**	64.24



(a) Categorizing the relations on which each method has the best result



(b) Break-down of methods achieving best performance on each type of relations

Figure 3.7: Models with best `FMRR`$^{\uparrow}$ on YAGO3-10

Table 3.11: Link prediction results on YAGO3-10

	YAGO3-10				YAGO3-10-DR			
Model	FH1$^{\uparrow}$	FMR$^{\downarrow}$	FH10$^{\uparrow}$	FMRR$^{\uparrow}$	FH1$^{\uparrow}$	FMR$^{\downarrow}$	FH10$^{\uparrow}$	FMRR$^{\uparrow}$
TransE	– 40.7	– 1193.4	– 67.9	– 0.504	– 11.7	– 2355.9	– 32.3	– 0.19
DistMult	– 34.2 42.4	– 1712.8 2685.1	– 64.9 66.0	– 0.448 0.51	– 9.6 13.3	– 7509.3 5553.2	– 28.8 30.7	– 0.161 0.192
ComplEx	– 35.5 44.9	– 3076.4 2911.7	– 64.6 67.8	– 0.455 0.53	– 9.7 14.3	– 8498.1 6077	– 28.8 31.5	– 0.162 0.201
ConvE	45.0 46.2	2792 1598.8	66.0 68.6	0.52 0.542	– 14.7	– 4453.3	– 31.5	– 0.204
RotatE	40.2 40.5	1767 1809.4	67.0 67.4	0.495 0.499	– 15.3	– 3084.2	– 33.2	– 0.214
TuckER	– 40.7	– 2293.8	– 66.1	– 0.496	– 14.8	– 6068.8	– 32	– 0.207
AMIE	55.8	24133	57.96	0.565	–	–	–	–

Table 3.12: `FHits@1`$^{\uparrow}$ results

Model	**FB15k**	**FB15k-237**	**WN18**	**WN18RR**
TransE	– 26.9	– 19.1	– 31.1	– 5.1
DistMult	– – 54.1	– 15.5 –	– – 75.2	– 29.1 –
ComplEX	59.9 – 59.5	– 15.9 –	93.6 – 93.7	– 34.0 –
ConvE	67.0 60.7	23.9 23.13	93.5 93.9	39.0 39.2
RotatE	74.6 73.8	24.1 23.9	94.4 94.4	42.8 42.5
TuckER	74.1 72.9	26.6 26.2	94.9 94.6	44.3 42.8
AMIE	– 75.1	– 22.5	– 93.9	– 35.6
Simple Model (generated by us)	71.6	1.1	96.4	34.8

CHAPTER 4

Inadequacy of Evaluation Metrics

In this chapter, we investigate and demonstrate the inadequacy of existing evaluation metrics. The frequently used ranking based metrics discussed in Section 2.3.2 have flaws in measuring the performance of the models. These flaws are due to closed-world assumption and aggregation of different prediction tasks. We will explain these problems and show how they impact the results. We also provide the results of type filtering – removing type inconsistent results from the ranked list.

To conduct the experiments in this chapter, we used the LibKGE library [47]. [1] Ruffinelli et al. [48] conducted an extensive experimental study to find the impact of hyperparameter optimization and training strategies on the performance of the models. They found that these factors significantly affect models' progress and could be as crucial as models' architecture. They provided pretrained models that were generated using a common experiment setup with extensive hyperparameter tuning as part of their open source LibKGE framework. Hence, to alleviate the impact of different training strategies in our experiments, we used the pretrained models from LibKGE.

4.1 Colsed-World Assumption

As it is briefly explained in Section 3.3, the performance measures in Section 2.3.2 are based on the closed-world assumption in which unknown facts are considered false. The terms open- and closed-world assumptions were first introduced by Reiter [29, 49]. Sup-

[1]`https://github.com/uma-pi1/kge`

pose that the total knowledge of a domain is available. Then it is not necessary to explicitly represent negative facts; they are inferred from the absence of their positive counterparts. Reiter [29] called this the closed-world assumption. It can be stated as "if P is not provable from the knowledge base, assume $\neg P$", where P is a positive fact. As mentioned earlier, this assumption is far from realistic due to the incompleteness of general real-world knowledge graphs and thus have flaws when a model correctly predicts a triple that does not exist in the benchmark dataset. More specifically, if a corrupted triple of a given test triple does not exist in the current dataset, it is considered incorrect. However, the corrupted triple might be correct as well. If a model ranks the corrupted triple higher than the test triple itself, its accuracy measures will be penalized, which contradicts the exact goal of link prediction—finding correct triples that do not already exist in the dataset [1, 50, 24].

As it was explained in Section 2.3.2, to calculate the filtered measures, corrupted triples that are already in training, test, or validation sets do not participate in the ranking. In this way, a model is not penalized for ranking other correct triples higher than a test triple. However, there might still be some other correct triples in the ranking list that do not exist in the benchmark dataset. This section used the original Freebase knowledge graph from Section 3.1 to check whether the predicted triples not present in the current dataset exist in the Freebase or not and filtered the correct responses to attain a more realistic evaluation of the models. Freebase is more comprehensive than FB15k-237, so this approach can verify whether models are penalized by closed-world assumptions. However, we should notice Freebase is still far from complete, and we cannot determine the true performance of the models by using it to calculate the measures.

Table 4.1 shows the obtained results. As it can be seen, using Freebase to filter out the correct predictions from the ranked list led to better performance, and models' FMRR$^{\uparrow}$ has improved by various degrees, from 5.75% to 7%, for different models. This proves that

Table 4.1: Results after using FB15k-237 vs. Freebase to filter out the correct predictions from the ranked list

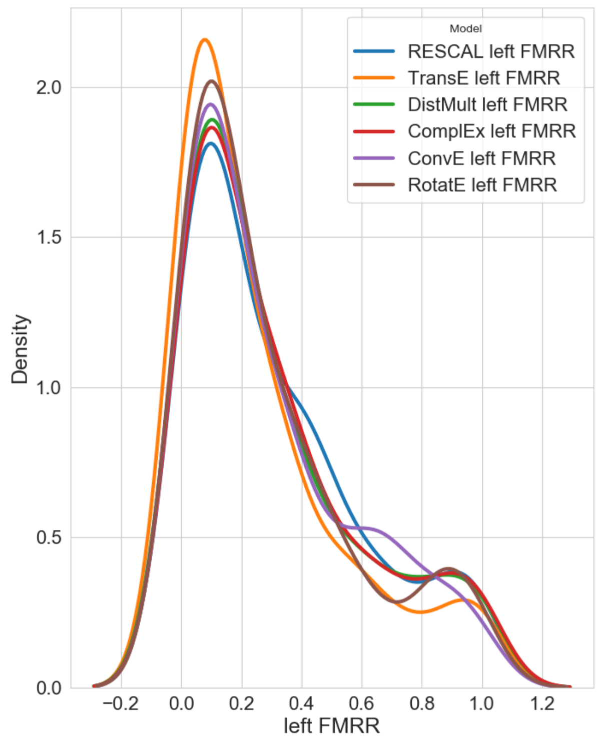
	FB15k-237				**Freebase**			
Model	FMRR↑	FH1↑	FH3↑	FH10↑	FMRR↑	FH1↑	FH3↑	FH10↑
RESCAL	0.356	0.263	0.393	0.541	0.378 (6.18%)	0.289	0.415	0.553
TransE	0.313	0.221	0.347	0.497	0.331 (5.75%)	0.241	0.366	0.508
DistMult	0.343	0.250	0.378	0.531	0.367 (7%)	0.278	0.403	0.544
ComplEx	0.348	0.253	0.384	0.536	0.372 (6.9%)	0.281	0.408	0.547
ConvE	0.339	0.248	0.369	0.521	0.360 (6.19%)	0.273	0.392	0.532
RotatE	0.333	0.240	0.368	0.522	0.355 (6.61%)	0.264	0.388	0.533

models are penalized for predicting correct responses, and metrics based on closed-world assumption underestimated models' accuracy.

Addressing this problem is challenging, because ideally we will need an absolute ground-truth dataset which entails the impossible work of manually labeling all possible predictions, given any large-scale knowledge graph. Furthermore, such a ground-truth dataset will not require completion, from a practical viewpoint.

4.2 Global Metrics

Another limitation of the current metrics is that they aggregate the predictions' accuracy of all triples into a single value which makes it impossible to discern the specific strengths and weaknesses of the models. We suggest computing the left and right entity link prediction results per relation separately and comparing models by analyzing the distribution of those results instead of examining the overall average of test triples' prediction accuracy. Figures 4.1 and 4.2 as well as Figures 4.3a and 4.3b show such results on FB15k-237. We can observe that left and right entity predictions ((?, r, t) and (h, r, ?), respectively)



(a) Density plot

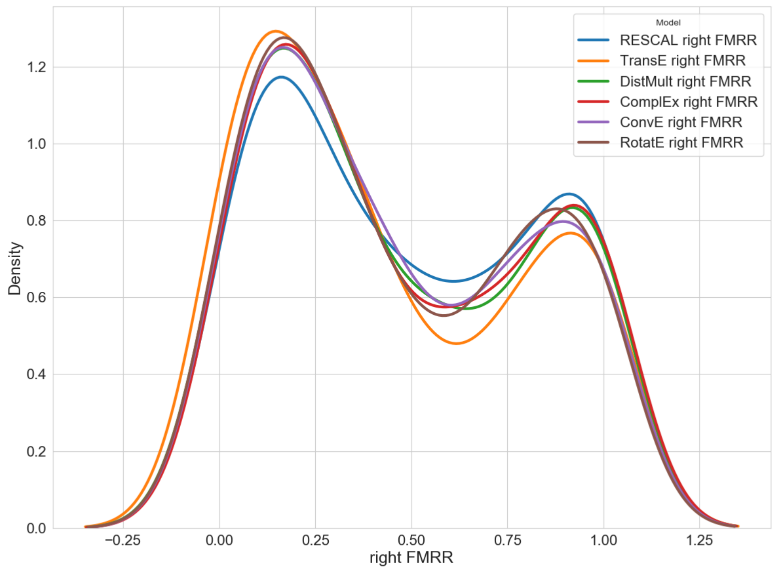
Model	Left $\texttt{FMRR}^{\uparrow}$
RESCAL	0.260
TransE	0.208
DistMult	0.246
ComplEx	0.250
ConvE	0.239
RotatE	0.229

(b) Average

Figure 4.1: Density plot of left $\texttt{FMRR}^{\uparrow}$ per relation vs. the average of left $\texttt{FMRR}^{\uparrow}$ for all test triples on FB15k-237

differ considerably on FB15k-237. Models have higher accuracy on right entity predictions than on left entity predictions. The left and right results clearly have different distributions. The distribution of the right results is bimodal and has two peaks, one peak around 0.1 and another one around 1 while the left distribution has one peak around 0.1.

Yet another problem of the global metrics is that they measure the micro average of all predictions' accuracy. Figures 4.4a and 4.4b show the frequency of relations in training and test set of FB15k-237. As it can be observed, this dataset is highly skewed with a few relations of high frequency and a large number of relations with low frequency. Hence, we suggest using the macro average of per-relation accuracy, in order to treat all relations equally in evaluation. Table 4.2 compares the results of micro averaging vs. macro av-



(a) Density plot

Model	Right FMRR$^{\uparrow}$
RESCAL	0.452
TransE	0.418
DistMult	0.439
ComplEx	0.445
ConvE	0.439
RotatE	0.438

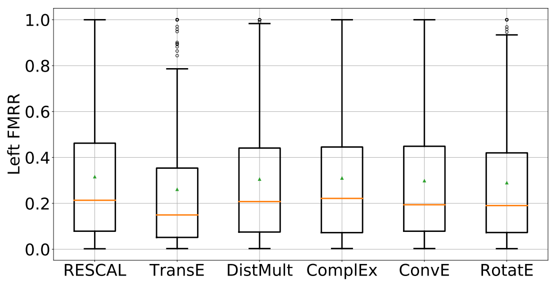
(b) Average

Figure 4.2: Density plot of right FMRR$^{\uparrow}$ per relation vs. the average of right FMRR$^{\uparrow}$ for all test triples on FB15k-237

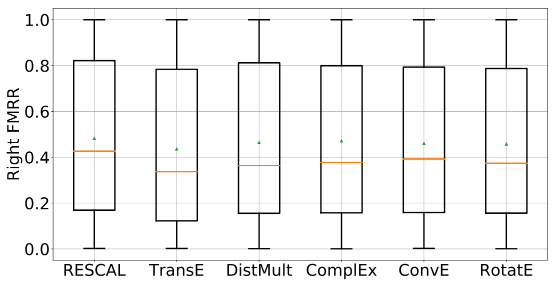
eraging for left and right FMRR$^{\uparrow}$. As observed, FMRR$^{\uparrow}$ of all models improves when macro averaging is used, which shows there are low-frequency relations in the dataset on which the models have high accuracy.

4.3 Type Filtering

Freebase has millions of topics about real-world entities like people, places, and things. Each entity in Freebase has one or more types [27]. For example Bob Dylan has types such as singer, performer, book author, and actor. Types themselves are grouped into domains. For example, the Actor type is in the Film domain. Type information is stated by a special relation *object/type* in Freebase, e.g., (Bob Dylan, *object/type*, singer). We can extract these type and domain definitions from Freebase and use them to filter out the type-inconsistent predictions. For example, when we want to predict the University which a



(a) Left FMRR$^{\uparrow}$



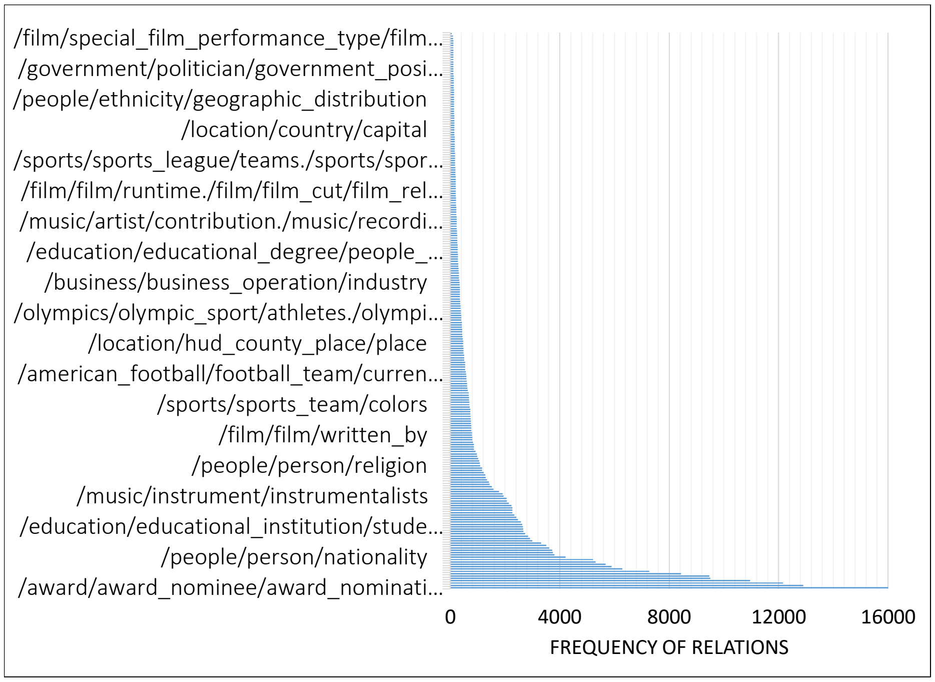
(b) Right FMRR$^{\uparrow}$

Figure 4.3: Box Plot of FMRR$^{\uparrow}$ per relation on FB15k-237 with multiple models

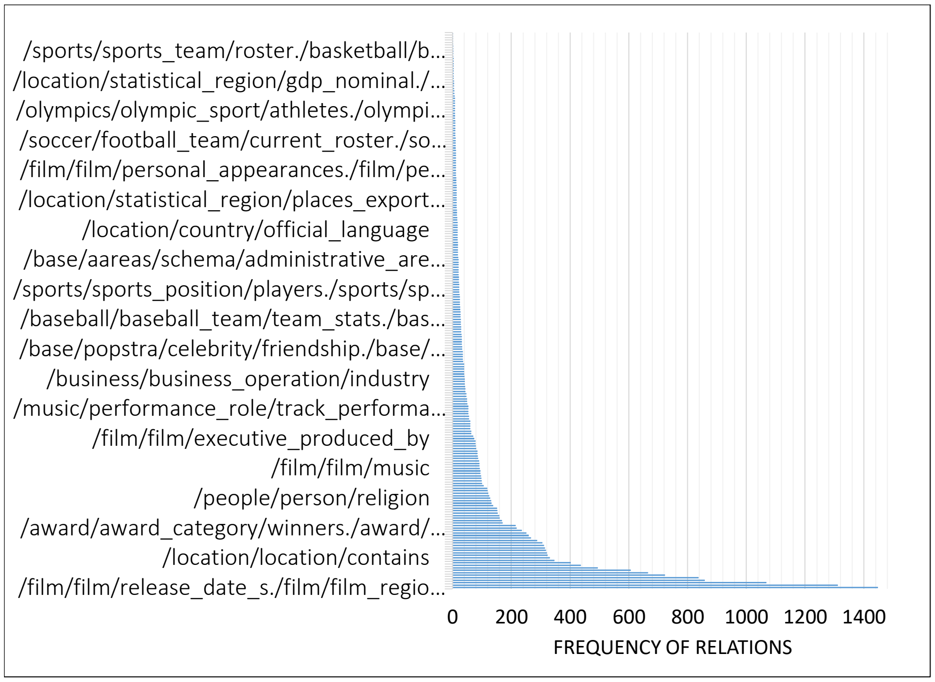
Table 4.2: Micro averaging vs. macro averaging

	Micro average		**Macro average**	
Model	Left FMRR$^{\uparrow}$	Right FMRR$^{\uparrow}$	Left FMRR$^{\uparrow}$	Right FMRR$^{\uparrow}$
RESCAL	0.260	0.452	0.316	0.483
TransE	0.208	0.418	0.261	0.436
DistMult	0.246	0.439	0.305	0.465
ComplEx	0.250	0.445	0.310	0.472
ConvE	0.239	0.439	0.298	0.461
RotatE	0.229	0.438	0.290	0.458

person is graduated from, we can only rank the entities with the type University. This section provides link prediction results on FB15k-237 after removing type inconsistent entities from the ranked list of predictions. Table 4.3 shows a slight improvement in the results after conducting type filtering. It proves that there are a few type inconsistent predictions in the ranked list placed before the test triple. However, it seems that models are usually able to capture entity types.



(a) Training set of FB15k-237



(b) Test set of FB15k-237

Figure 4.4: Frequency of relations in FB15k-237

Table 4.3: Type filtering results

	Original Results (LibKGE)				Type Filtering			
Model	`FMRR`$^{\uparrow}$	`FH1`$^{\uparrow}$	`FH3`$^{\uparrow}$	`FH10`$^{\uparrow}$	`FMRR`$^{\uparrow}$	`FH1`$^{\uparrow}$	`FH3`$^{\uparrow}$	`FH10`$^{\uparrow}$
RESCAL	0.356	0.263	0.393	0.541	0.357	0.263	0.393	0.541
TransE	0.313	0.221	0.347	0.497	0.317	0.226	0.351	0.5
DistMult	0.343	0.25	0.378	0.531	0.344	0.249	0.38	0.532
ComplEx	0.348	0.253	0.384	0.536	0.349	0.254	0.385	0.537
ConvE	0.339	0.248	0.369	0.521	0.339	0.248	0.37	0.522
RotatE	0.333	0.24	0.368	0.522	0.336	0.242	0.37	0.524

CHAPTER 5

Problems of the Generic Evaluation Protocol (Link Prediction)

In this chapter, we discuss the problems associated with link prediction as a widely-used evaluation protocol employed to measure the performance of the embedding models. We also provide the results of some alternative evaluation protocols, such as entity-pair ranking, property prediction, and triple classification. The performance of the models on these evaluation protocols is unsatisfactory. There is also a mismatch between link prediction results and performance on these other tasks. Based on results of experiments conducted using the LibKGE library, RESCAL has the strongest link prediction performance. However, on entity-pair ranking and property prediction, RotatE is the best performing model; DistMult and ComplEx have better performance on triple classification. The evaluation results using these protocols suggest that better knowledge graph completion methods and training strategies are needed.

5.1 Entity-Pair Ranking

Link prediction, the most generic knowledge graph embedding evaluation protocol, verifies that models prioritize correct answers over wrong ones for a question that is already known to have an answer. This evaluation setup can be misleading as we cannot verify if a model ranks false or nonsensical triples lower than correct triples. For example, consider a test triple (James Ivory, *director/film*, A Room With A View). Evaluation by link prediction

Table 5.1: Pair-ranking results on FB15K-237. Results in blue color are taken from [1].

Model	MAP@100$^{\uparrow}$	WMAP@100$^{\uparrow}$	P@100$^{\uparrow}$	WP@100$^{\uparrow}$
RESCAL	– 0.048	0.067 0.070	– 6.1	15.0 11.1
TransE	– 0.043	0.079 0.054	– 5.1	17.6 9.1
DistMult	– 0.003	0.030 0.002	– 1.1	4.2 1.4
ComplEx	– 0.062	0.071 0.073	– 6.8	16.6 11.4
ConvE	– 0.017	– 0.021	– 2.3	– 3.8
RotatE	– 0.067	– 0.069	– 5.5	– 8.9

only seeks responses to two sensible questions: "who directed the movie A Room With A View?", i.e., (?, *director/film*, A Room With A View), and "which movie is directed by James Ivory?", i.e., (James Ivory, *director/film*, ?). It does not verify if a model would rank false or nonsensical triples such as (A Room With A View, *director/film*, James Ivory) lower than the correct one. Wang et al. [1] discussed this problem and proposed to use an alternative protocol called entity-pair ranking to rank all possible triples for a specific relation. Following this evaluation protocol, we would create all possible combinations of entities for a relation and then rank created triples based on their scores. To measure a model's performance, Wang et al. [1] proposed using weighted Mean Average Precision (WMAP@K$^{\uparrow}$) and weighted precision (WP@K$^{\uparrow}$). These two metrics are defined as:

$$WMAP@K = \sum_{r \in \mathcal{R}} AP_r@K \times \frac{min(K, |T_r|)}{\sum_{r' \in \mathcal{R}} min(K, |T_{r'}|)} \tag{5.1}$$

Table 5.2: Pair-ranking results on WN18RR. Results in blue color are taken from [1].

Model	`MAP@100`$^{\uparrow}$	`WMAP@100`$^{\uparrow}$	`P@100`$^{\uparrow}$	`WP@100`$^{\uparrow}$
RESCAL	– 0.191	0.131 0.225	– 20.1	13.8 26.1
TransE	– 0.003	0.020 0.004	– 1.9	1.3 2.8
DistMult	– 0.249	0.141 0.170	– 12.6	17.8 15.1
ComplEx	– 0.228	0.168 0.170	– 17.5	20.0 21.8
ConvE	– 0.041	– 0.060	– 6.0	– 8.8
RotatE	– 0.172	– 0.121	– 15.4	– 21.1

where $AP_r@K$ is the average precision of the top K predictions for relation r and $|T_r|$ represents the number of test triples for r, and

$$WP@K = \sum_{r\in\mathcal{R}} P_r@K \times \frac{min(K,|T_r|)}{\sum_{r'\in\mathcal{R}} min(K,|T_{r'}|)} \tag{5.2}$$

where $P_r@K$ is the fraction of correct predictions among top K predictions for relation r.

We used the same metrics to measure the performance of the models. However, we report the unweighted version, which is equivalent to macro averaging, treating each relation equally. We define $AP_r@K$ and $MAP@K$ as follows:

$$\begin{aligned} AP_r@K &= \frac{1}{NP_r}\sum_{k\in K} P_r@k \times relv_r[k], \\ MAP@K &= \sum_{r\in\mathcal{R}} AP_r@K \end{aligned} \tag{5.3}$$

where NP_r is minimum of K and $|T_r|$, $P_r@k$ is the precision among top k predictions, and $rel_r[k]$ represents whether the kth prediction is correct or not. It will be 1 if the prediction is correct and 0 otherwise. Also, $P@K$ is defined as:

$$P@K = \sum_{r \in \mathcal{R}} P_r@K \tag{5.4}$$

We used the LibKGE pretrained models [47] and evaluated the models by the pair-ranking protocol. Tables 5.1 and 5.2 show the results. The overall observation is that the models have low performance on entity-pair ranking. DistMult and ConvE are the worst-performing models on FB15k-237, while ComplEx has the best performance. On WN18-RR, RESCAL outperforms other models, while TransE has the lowest accuracy. All the relations with $\texttt{AP}_r\texttt{@100}^{\uparrow}$ higher than 0.5 on FB15k-237 for each model can be found in Tables 5.3- 5.7. "# of rel" in tables indicates the number of test triples with a specific relation. When using DistMult on entity-pair ranking, the $\texttt{AP}_r\texttt{@100}^{\uparrow}$ for all relations is less than 0.05. Therefore, we are not showing such a table for DistMult. As the tables show, some of the relations with the best $\texttt{AP}_r\texttt{@100}^{\uparrow}$ are Cartesian product relations, e.g., */olympic_games/medals _awarded./olympic_medal_honor/medal* and */travel_destination/climate./travel_destination_monthly _climate/month*. Table 5.8 and Figure 5.1 show the $\texttt{AP}_r\texttt{@100}^{\uparrow}$ of each relation in WN18-RR for all the models. The models have high performance on 3 symmetric relations #1 (*_derivationally_related_form*), #2 (*_similar_to*), and #3 (*_verb_group*), while on other relations their accuracy is zero or close to zero.

5.2 Property Prediction

The current evaluation protocol of link prediction is based on the assumption that the presence of a particular property on an entity is already known. The evaluation focuses on whether a model can derive the correct property values. In reality, though, it remains a challenge to determine whether a property is valid for a given entity in the first place.

Table 5.3: RESCAL best results

Relations with $\mathtt{AP}_r\mathtt{@100}^{\uparrow}$ greater than 0.5			
Relation	**# of rel**	$\mathtt{AP}_r\mathtt{@100}^{\uparrow}$	$\mathtt{P}_r\mathtt{@100}^{\uparrow}$
/person/gender	436	0.95	0.98
/event/instance_of_recurring_event	11	0.82	0.09
/award_category/winners./award_honor/ceremony	323	0.75	0.88
/person/nationality	494	0.64	0.8
/person/profession	1311	0.55	0.73

Table 5.4: TransE best results

Relations with $\mathtt{AP}_r\mathtt{@100}^{\uparrow}$ greater than 0.5			
Relation	**# of rel**	$\mathtt{AP}_r\mathtt{@100}^{\uparrow}$	$\mathtt{P}_r\mathtt{@100}^{\uparrow}$
/olympic_games/medals_awarded./olympic_medal _honor/medal	16	1	0.16
/film/language	314	0.86	0.92
/event/instance_of_recurring_event	11	0.80	0.09
/food/nutrients./nutrition_fact/nutrient	105	0.56	0.66

Therefore, we propose the property prediction task as another evaluation protocol. For conducting this task, we have changed the test set of FB15k-237. We just need (h, r) pairs in the test set. We kept a pair (h, r) in test data if triples of the form (h, r, t') are not available in the training set— since we want to predict the property r for entity h, we must make sure this information is not already available in the training set. After this procedure, there are 4,755 test cases of (h, r) with 3,825 distinct entities and 163 distinct relations in the test set. The test set of WN18RR is changed in the same way, and 1,140 instances of (h, r) with 1,136 entities and 9 relations remained. To conduct property prediction, given a test case (h_1,r_1), we calculate the score of triples (h_1, r', t') where $\mathsf{t}' \in \mathcal{E}$ and $r' \in \mathcal{R}$. Then, the triple

Table 5.5: ComplEx best results

Relations with $\texttt{AP}_r\texttt{@100}^{\uparrow}$ greater than 0.5			
Relation	**# of rel**	$\texttt{AP}_r\texttt{@100}^{\uparrow}$	$\texttt{P}_r\texttt{@100}^{\uparrow}$
/event/instance_of_recurring_event	11	0.80	0.09
/olympic_games/medals_awarded./olympic_medal _honor/medal	16	0.75	0.16
/person/profession	1311	0.73	0.84
/award_category/category_of	20	0.63	0.15
/tv_program/languages	24	0.63	0.22
/person/nationality	494	0.61	0.77
/non_profit_organization/registered_with./non_profit _registration/registering_agency	22	0.61	0.19
/aareas/schema/administrative_area/administrative _area_type	20	0.59	0.2
/baseball_team/team_stats./baseball_team _stats/season	27	0.56	0.26
/politician/government_positions_held./government _position_held/legislative_sessions	30	0.52	0.25
/food/nutrients./nutrition_fact/nutrient	105	0.50	0.73

with the highest score is saved. Finally, the relation of the triple with the highest score is selected as the property of the entity h_1.

The results of Table 5.10 show that models have deficient performance on the task of property prediction. The $\texttt{FMRR}^{\uparrow}$ of the best performing model, RotatE, is 0.182. It indicates that models are mainly unable to predict the property of an entity. The overall performance of the models on WN18RR is considerably better than results on FB15K-237. However, we should notice that the number of candidate properties in WN18RR is 11, while it is 237 in FB15K-237.

Table 5.6: ConvE best results

Relations with `AP`$_r$`@100`$^{\uparrow}$ **greater than 0.5**			
Relation	**# of rel**	`AP`$_r$`@100`$^{\uparrow}$	`P`$_r$`@100`$^{\uparrow}$
/food/nutrients./nutrition_fact/nutrient	105	0.75	0.8

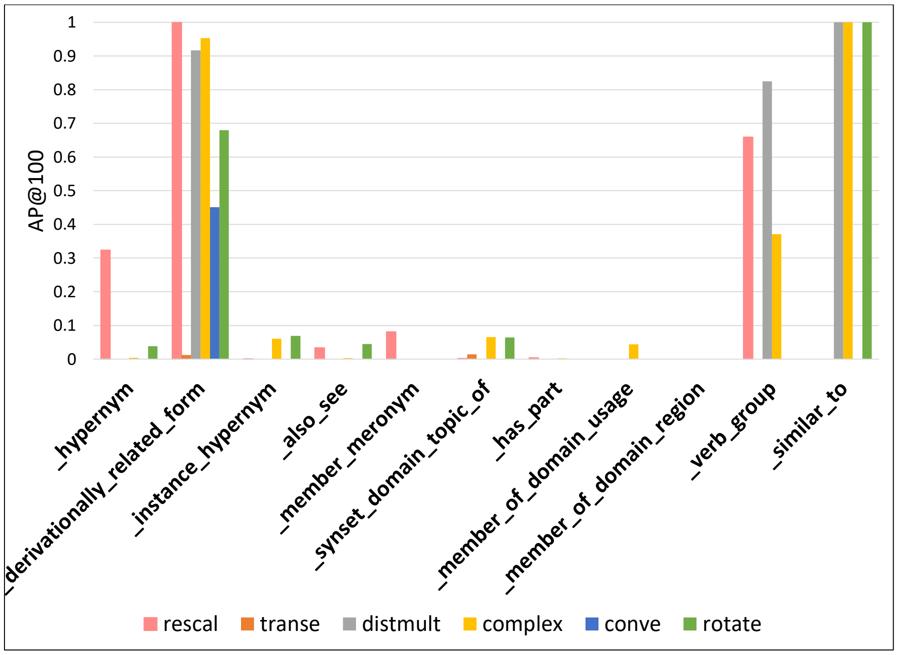


Figure 5.1: `AP`$_r$`@100`$^{\uparrow}$ of each relation of WN18-RR

5.3 Triple Classification

Another way of evaluation is to find the models' performance on triple classification. This task is the binary classification of triples regarding whether they are true or false facts. To conduct triple classification, we need to create negative samples for the test and validation sets and then learn a classification threshold for each relation on validation data. After learning the thresholds, each test triple with a score higher than the specific threshold for its relation is considered positive and negative otherwise. Some of the earlier knowledge

Table 5.7: RotatE best results

Relations with $\mathtt{AP}_r\mathtt{@100}^{\uparrow}$ greater than 0.5			
Relation	**# of rel**	$\mathtt{AP}_r\mathtt{@100}^{\uparrow}$	$\mathtt{P}_r\mathtt{@100}^{\uparrow}$
/olympic_games/medals_awarded./olympic_medal _honor/medal	16	1	0.16
/aareas/schema/administrative_area/administrative _area_type	20	0.93	0.2
/travel_destination/climate./travel_destination _monthly_climate/month	60	0.91	0.57
/event/instance_of_recurring_event	11	0.82	0.09
/non_profit_organization/registered_with./non_profit _registration/registering_agency	22	0.82	0.18
/petbreeds/city_with_dogs/top _breeds./petbreeds/dog_city_relationship/dog_breed	21	0.70	0.21
/film/language	314	0.62	0.77
/olympic_participating_country/medals _won./olympic_medal_honor/medal	38	0.57	0.3

graph embedding models were evaluated using triple classification [30, 31]. However, their negative triples were generated by randomly corrupting head or tail entities of test and validation triples. These randomly generated negative test cases are not challenging, leading to overestimated classification accuracy. Pezeshkpour et al. [51] and Safavi et al. [52] noted this problem and created some hard negative samples for their generated benchmark datasets. Inspired by their work, we created two sets of negative samples for test and validation sets of FB15K-237. One set complies with type constraints and the other violates such constraints. As it was explained in Section 4.3, each entity in Freebase has one or more types and these types are denoted by the special relation *object/type*. The types of entities were extracted from Freebase using this relation. Then, we used the extracted

Table 5.8: Entity-pair ranking results for each relation of WN18-RR using $\texttt{AP}_r\texttt{@100}^{\uparrow}$

Relation	**# of rel**	**RESCAL**	**TransE**	**DistMult**	**ComplEx**	**ConvE**	**RotatE**
r1	1251	0.324	0	0	0.004	0	0.038
r2	1074	1	0.013	0.917	0.953	0.451	0.680
r3	122	0	0	0	0.061	0	0.069
r4	56	0.034	0	0	0.003	0	0.045
r5	253	0.081	0	0	0	0	0
r6	114	0.002	0.014	0	0.065	0	0.064
r7	172	0.004	0	0	0.002	0	0
r8	24	0	0	0	0.044	0	0
r9	26	0	0	0	0	0	0
r10	39	0.659	0	0.824	0.371	0	0
r11	3	0	0	1	1	0	1

Table 5.9: Relations used in Table 5.8

r1	*_hypernym*
r2	*_derivationally_related_form*
r3	*_instance_hypernym*
r4	*_also_see*
r5	*_member_meronym*
r6	*_synset_domain_topic_of*
r7	*_has_part*
r8	*_member_of_domain_usage*
r9	*_member_of_domain_region*
r10	*_verb_group*
r11	*_similar_to*

Table 5.10: Property prediction results on FB15K-237

Model	FMR$^{\downarrow}$	FMRR$^{\uparrow}$
RESCAL	41.99	0.028
TransE	145.8	0.019
DistMult	23.1	0.126
ComplEx	22.89	0.131
ConvE	44.72	0.161
RotatE	**18.21**	**0.182**

Table 5.11: Property prediction results on WN18RR

Model	FMR$^{\downarrow}$	FMRR$^{\uparrow}$
RESCAL	2.48	0.592
TransE	5.82	0.198
DistMult	3.43	0.492
ComplEx	2.79	0.610
ConvE	4.75	0.400
RotatE	**1.90**	**0.740**

type information and the ranked list of predictions generated by an embedding model for tail or head entity link prediction to generate both of the aforementioned sets. To generate a type consistent negative triple for a test triple (h, r, t), we scan the ranked list generated for tail entity prediction to find the first entity t' in the list that has the same type as t. We then verify that the corrupted triple (h, r, t') is negative by checking whether it exists in the labeled dataset FB15K-237 as well as the original Freebase snapshot from Section 3.1. If it does not exist in any of them, triple (h, r, t') is added to the set of type consistent negative triples for tail entities. We repeat the same procedure and create another set of

Table 5.12: Triple classification results

	consistent h				**inconsistent** h			
Model	Precision	Recall	Acc	F1	Precision	Recall	Acc	F1
RESCAL	0.59	0.37	0.55	0.45	0.95	0.83	0.89	0.89
TransE	0.52	0.59	0.52	0.55	0.81	0.69	0.76	0.74
DistMult	0.53	0.51	0.53	0.52	0.94	0.87	0.91	0.90
ComplEx	0.54	0.48	0.53	0.51	0.94	0.88	0.91	0.91
ConvE	0.54	0.53	0.54	0.53	0.57	0.72	0.59	0.64
RotatE	0.52	0.53	0.52	0.52	0.89	0.83	0.87	0.86

	consistent t				**inconsistent** t			
Model	Precision	Recall	Acc	F1	Precision	Recall	Acc	F1
RESCAL	0.64	0.45	0.60	0.53	0.95	0.86	0.91	0.90
TransE	0.58	0.54	0.57	0.56	0.90	0.82	0.86	0.86
DistMult	0.59	0.55	0.58	0.57	0.95	0.89	0.92	0.92
ComplEx	0.60	0.56	0.59	0.58	0.95	0.90	0.93	0.92
ConvE	0.62	0.41	0.58	0.49	0.95	0.83	0.89	0.88
RotatE	0.60	0.47	0.58	0.53	0.87	0.78	0.83	0.82

type consistent negative triples by corrupting the head entities. The same procedure is used to create negative samples for validation data. To generate type-violating negative triples we just make sure that the type of the entity that is used to corrupt a positive triple is different from the original entity's type.

The results of triple classification on all these new test sets are presented in Table 5.12. The first observation is that the models have low performance on type-consistent negative samples while their performance on type-violating ones is satisfactory. The scores of positive and negative triples differ clearly when negative samples are type violating. Another observation is that the models' precision is higher than their recall. As in the case of knowledge graphs, we care more about avoiding false positives and thus the lower recall of the models may be of less concern.

CHAPTER 6

Conclusions

In this dissertation, we extensively investigated data redundancy in the widely-used benchmark datasets FB15k, WN18, and YAGO3-10 and its impact on the performance of link prediction models. The majority of triples in these datasets form reverse or duplicate pairs on which link prediction task is trivial. Our experiments demonstrate that, in the absence of straightforward prediction tasks, the performance of the embedding models degenerates significantly. The results show that state-of-the-art embedding models lack sufficient performance to be deployed in a truly automated setting for conducting link prediction. Moreover, given the data characteristics, a simple rule-based model derived from data statistics can often challenge the accuracy of complex machine learning models.

A more fundamental defect of these models is that the link prediction scenario, given such data, is nonexistent in the real world. In Freebase, which was used to produce FB15k, when a new fact is inserted, it would be added as a pair of reverse triples. The relations of the two reverse triples are denoted explicitly by a particular relation in Freebase called *reverse_property* [27, 28]. For such reverse relations always curated into the datasets as pairs, there is never a scenario in which one needs to predict a triple while its reverse is already in the knowledge graph. This is a form of *overfitting* as the learned model is optimized for the reverse triples and cannot be generalized to realistic settings. More precisely, this is a case of excessive *data leakage*, and the models are tested on data that is already observed in the training data in some capacity. Thus the accuracy of the models is inflated.

We identified Cartesian product relations in FB15k, which also lead to unrealistic evaluation performance. Given such a relation, there are a set of subjects and a set of objects, and the relation is valid from every subject in the first set to every object in the second set. Cartesian product relations in FB15k are due to the concatenation of edges on CVT nodes and present unrealistic cases of link prediction. Moreover, if the goal is to do link prediction on such relations, a straightforward model could be employed instead of embedding models. We can view a relation as a bipartite graph between its subjects and objects. The simple method declares a relation as a Cartesian product if the graph is close to a complete bipartite graph based on a threshold. For link prediction, the simple method considers every edge in the complete bipartite graph to be true.

We also demonstrated the inadequacy of existing evaluation metrics that penalize a method for generating correct predictions not available in the labeled dataset. The commonly-used evaluation metrics are based on the closed-world assumption. Under the closed-world assumption, non-observed triples are considered wrong. Thus if a model's predictions are correct but do not exist in the benchmark dataset, then the model's accuracy may decrease. We used a May 2013 snapshot of Freebase to verify whether a model's predictions are correct and observed improvement in the evaluation measures. We also showed that aggregating a model's performance on all triples into a single accuracy value will hide specific strengths and weaknesses of the models. We presented detailed results of models, calculated performance per relation, demonstrated results distribution and results on relations of different levels of difficulty to provide more insights into embedding models.

As link prediction, the most generic evaluation protocol, has different issues, we reported the performance of the models on some other protocols called entity-pair ranking, property prediction, and triple classification. The performance of the models using all these evaluation protocols is unsatisfactory. It suggests that better knowledge graph embedding models or training strategies are needed.

REFERENCES

[1] Y. Wang, D. Ruffinelli, R. Gemulla, S. Broscheit, and C. Meilicke, "On evaluating embedding models for knowledge base completion," in *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pp. 104–112, 2019.

[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM international conference on Management of data (SIGMOD)*, pp. 1247–1250, 2008.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, "Dbpedia: A nucleus for a web of open data," in *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC + ASWC)*, pp. 722–735, 2007.

[4] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell, "Toward an architecture for never-ending language learning," in *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1306–1313, 2010.

[5] D. Vrandečić and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[6] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: A large ontology from wikipedia and wordnet," *Web Semantics*, vol. 6, no. 3, pp. 203–217, 2008.

[7] X. Yao and B. Van Durme, "Information extraction over structured data: Question answering with freebase," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, vol. 1, pp. 956–966, 2014.

[8] W.-t. Yih, M.-W. Chang, X. He, and J. Gao, “Semantic parsing via staged query graph generation: Question answering with knowledge base,” in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL + IJCNLP)*, pp. 1321–1331, July 2015.

[9] K. Xu, S. Reddy, Y. Feng, S. Huang, and D. Zhao, “Question answering on freebase via relation extraction and textual evidence,” in *In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 2326–2336, Aug. 2016.

[10] J. S. Eder, “Knowledge graph based search system,” June 21 2012. US Patent App. 13/404,109.

[11] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, and Q. He, “A survey on knowledge graph-based recommender systems,” *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[12] M. Rotmensch, Y. Halpern, A. Tlimat, S. Horng, and D. Alexander Sontag, “Learning a health knowledge graph from electronic medical records,” *Scientific Reports*, vol. 7, 12 2017.

[13] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, “A review of relational machine learning for knowledge graphs,” *Proceedings of the IEEE*, vol. 104, no. 1, pp. 11–33, 2016.

[14] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, and O. Yakhnenko, “Translating embeddings for modeling multi-relational data,” in *Proceedings of the 26th International Conference on Neural Information Processing Systems (NeurIPS)*, pp. 2787–2795, 2013.

[15] M. Nickel, V. Tresp, and H.-P. Kriegel, “A three-way model for collective learning on multi-relational data,” in *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pp. 809–816, 2011.

[16] H. Cai, V. W. Zheng, and K. C.-C. Chang, “A comprehensive survey of graph embedding: Problems, techniques, and applications,” *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 30, no. 9, pp. 1616–1637, 2018.

[17] A. Rossi, D. Barbosa, D. Firmani, A. Matinata, and P. Merialdo, “Knowledge graph embedding for link prediction: A comparative analysis,” *ACM Trans. Knowl. Discov. Data*, vol. 15, Jan. 2021.

[18] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek, “Amie: association rule mining under incomplete evidence in ontological knowledge bases,” in *Proceedings of the 22nd international conference on World Wide Web (WWW)*, pp. 413–422, 2013.

[19] N. Lao and W. W. Cohen, “Relational retrieval using a combination of path-constrained random walks,” *Machine learning*, vol. 81, no. 1, pp. 53–67, 2010.

[20] G. A. Miller, “Wordnet: A lexical database for english,” *Communications of the ACM*, vol. 38, pp. 39–41, Nov. 1995.

[21] K. Toutanova and D. Chen, “Observed versus latent features for knowledge base and text inference,” in *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pp. 57–66, 2015.

[22] T. Dettmers, M. Pasquale, S. Pontus, and S. Riedel, “Convolutional 2d knowledge graph embeddings,” in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1811–1818, February 2018.

[23] F. Akrami, L. Guo, W. Hu, and C. Li, “Re-evaluating embedding-based knowledge graph completion methods,” in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 1779–1782, 2018.

[24] F. Akrami, M. S. Saeef, Q. Zhang, W. Hu, and C. Li, “Realistic re-evaluation of knowledge graph completion methods: An experimental study,” in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD ’20, (New York, NY, USA), p. 1995–2010, Association for Computing Machinery, 2020.

[25] I. Balažević, C. Allen, and T. M. Hospedales, “Tucker: Tensor factorization for knowledge graph completion,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP + IJCNLP)*, pp. 5185–5194, 2019.

[26] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, “Rotate: Knowledge graph embedding by relational rotation in complex space,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 926–934, 2019.

[27] T. Pellissier Tanon, D. Vrandečić, S. Schaffert, T. Steiner, and L. Pintscher, “From Freebase to Wikidata: The great migration,” in *Proceedings of the 25th International Conference on World Wide Web (WWW)*, pp. 1419–1428, 2016.

[28] M. Färber, *Semantic Search for Novel Information*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2017.

[29] R. Reiter, “On closed world data bases,” in *Gallaire H., Minker J. (eds) Logic and Data Bases*, pp. 55–76, 1978.

[30] R. Socher, D. Chen, C. D. Manning, and A. Ng, “Reasoning with neural tensor networks for knowledge base completion,” in *Proceedings of the 26th International Conference on Neural Information Processing Systems (NeurIPS)*, pp. 926–934, 2013.

[31] Z. Wang, J. Zhang, J. Feng, and Z. Chen, “Knowledge graph embedding by translating on hyperplanes,” in *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1112–1119, 2014.

[32] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion.," in *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 2181–2187, 2015.

[33] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL + IJCNLP)*, pp. 687–696, 2015.

[34] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[35] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pp. 2071–2080, 2016.

[36] L. R. Tucker *et al.*, "The extension of factor analysis to three-dimensional matrices," *Contributions to mathematical psychology*, vol. 110119, 1964.

[37] S. K. Mohamed, E. Muñoz, and V. Novacek, "On training knowledge graph embedding models," *Information*, vol. 12, no. 4, 2021.

[38] S. Muggleton and L. De Raedt, "Inductive logic programming: Theory and methods," *The Journal of Logic Programming*, vol. 19, pp. 629–679, 1994.

[39] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph, "OWL 2 web ontology language primer," *W3C recommendation*, vol. 27, no. 1, p. 123, 2009.

[40] F. Mahdisoltani, J. Biega, and F. M. Suchanek, "Yago3: A knowledge base from multilingual wikipedias," in *Conference on Innovative Data Systems Research (CIDR)*, 2015.

[41] Y. Lin, Z. Liu, H. Luan, M. Sun, S. Rao, and S. Liu, “Modeling relation paths for representation learning of knowledge bases,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 705–714, 2015.

[42] J. Weston, A. Bordes, O. Yakhnenko, and N. Usunier, “Connecting language and knowledge bases with embedding models for relation extraction,” in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1366–1371, 2013.

[43] B. Shi and T. Weninger, “Proje: Embedding projection for knowledge graph completion,” in *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, pp. 1236–1242, 2017.

[44] X. Han, S. Cao, X. Lv, Y. Lin, Z. Liu, M. Sun, and J. Li, “Openke: An open toolkit for knowledge embedding,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations (EMNLP)*, pp. 139–144, 2018.

[45] L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek, “Fast rule mining in ontological knowledge bases with amie++,” *The VLDB Journal*, vol. 24, pp. 707–730, Dec. 2015.

[46] C. Meilicke, M. Fink, Y. Wang, D. Ruffinelli, R. Gemulla, and H. Stuckenschmidt, “Fine-grained evaluation of rule-and embedding-based systems for knowledge graph completion,” in *International Semantic Web Conference (ISWC)*, 2018.

[47] S. Broscheit, D. Ruffinelli, A. Kochsiek, P. Betz, and R. Gemulla, “LibKGE - A knowledge graph embedding library for reproducible research,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 165–174, 2020.

[48] D. Ruffinelli, S. Broscheit, and R. Gemulla, “You CAN teach an old dog new tricks! on training knowledge graph embeddings,” in *International Conference on Learning Representations*, 2020.

[49] R. Reiter, “Deductive question-answering on relational data bases,” in *Logic and data bases*, pp. 149–177, Springer, 1978.

[50] T. Safavi, D. Koutra, and E. Meij, “Evaluating the Calibration of Knowledge Graph Embeddings for Trustworthy Link Prediction,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Online), pp. 8308–8321, Association for Computational Linguistics, Nov. 2020.

[51] P. Pezeshkpour, Y. Tian, and S. Singh, “Revisiting evaluation of knowledge base completion models,” in *Automated Knowledge Base Construction*, 2020.

[52] T. Safavi and D. Koutra, “CoDEx: A Comprehensive Knowledge Graph Completion Benchmark,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Online), pp. 8328–8350, Association for Computational Linguistics, Nov. 2020.