UNIVERSITY OF CALIFORNIA,
IRVINE

Investigating the logical inference capabilities of Knowledge Graph Embedding Models

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Computer Science

by

Chirag Choudhary

Thesis Committee:
Assistant Professor Sameer Singh, Chair
Assistant Professor Stephan Mandt
Professor Charless Fowlkes

2019

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I want to thank my advisor Dr. Sameer Singh for his continuous guidance and support throughout the completion of this thesis. He made me realize the importance of approaching my research with utmost sincerity, dedication and curiosity, and always encouraged me to think differently. His mentorship not only helped me to finish my thesis, but also helped me become a better researcher, and I'm extremely grateful for all his support.

I'm also thankful to my committee members, Professor Charless Fowlkes and Professor Stephan Mandt, for their valuable feedback on this work.

Secondly, I would like to thank my friends Archit, Chirag, Rajni, Anshuman, Casey, Nile, Preston, Abhishek and everyone else for all the advice (academic and otherwise) and encouragement.

Finally, I want to thank my family: my grandfather K.C. Choudhary, my parents Dr. Vineet Choudhary and Dr. Jaya Choudhary, my bua (aunt) Dr. Sangeeta Ahuja, my fufaji (uncle) Dr. Sanjeev Ahuja, and my siblings Parth, Ishita and Ishan.

ABSTRACT OF THE THESIS

Investigating the logical inference capabilities of Knowledge Graph Embedding Models

By

Chirag Choudhary

Master of Science in Computer Science

University of California, Irvine, 2019

Assistant Professor Sameer Singh, Chair

A knowledge graph represents factual information in the form of graphs, where nodes represent real-world entities such as people, places and movies, and edges represent the relationships between these entities. Existing knowledge graphs are far from complete. Knowledge graph completion or link prediction refers to the task of predicting new relations (links) between entities by deriving information from the existing relations. A number of link prediction model have been proposed, several of which make probabilistic predictions about new links. These models can be rule-based methods derived from observed edges, latent representation based embedding methods, or a combination of both. These methods must capture different kinds of relational patterns in the data, such as symmetry or inversion patterns to fully model the data. Rule-based methods explicitly learn these patterns, and provide an interpretable approach to predict new edges. With embedding based models, however, due to the nature of latent embeddings, it is difficult to understand what is being captured by these models. In this work, we explore the logical inference capabilities of knowledge graph embedding models. We experiment with various knowledge graph embedding models on synthetic datasets to identify specific properties of each model. The objective is to empirically validate the suitability of these models to learning different relational patterns that exist in real-world knowledge graphs.

Chapter 1

Introduction

Knowledge Graphs (KGs) are multi-relational knowledge bases structured as a graph, with entities (real-world objects or concepts) as nodes, and relations between them as labeled edges. These knowledge graphs are stored as a collection of triplets or facts = $\{(h, r, t)\}$, where h, t $\in \mathcal{E}$ and r $\in \mathcal{R}$, also referred to as RDF format. Here, $\mathcal{E}$ is the set of entities and $\mathcal{R}$ is the set of relations in the knowledge graph. Some examples of these large-scale knowledge graphs are YAGO [40], Freebase [2], DBPedia [1], WordNet [27], ConceptNet [26] and NELL [8]. These rich sources of relational information can be used in several important applications, including information retrieval, relational learning, entity linking, language modeling, question answering, and recommendation engines. Although these knowledge graphs can contain millions of entities and billions of facts, they nevertheless are still very much incomplete. A common task on knowledge graphs is to predict these missing links between entities, which is known as *link prediction* or *knowledge base completion*. Due to the huge size of these graphs, a significant amount of work has been done to predict these missing links automatically from the observed data. These models use the information present in the knowledge graph, and might incorporate additional external sources of information about the entities and relations present in a KG, such as textual occurrences of entities.

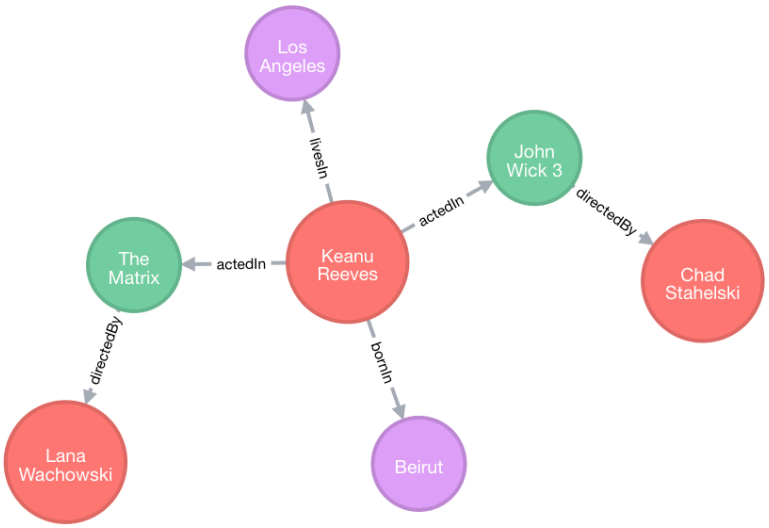


Figure 1.1: A Knowledge graph example with entities as nodes and relations as edges.

The traditional *Statistical Relational Learning* (SRL) methods use first-order logic for representing relational data, and probabilistic graphical models or inductive logic programming approaches for learning and inference. These methods can learn complex relational patterns from the data and are interpretable, but lack necessary scalability due to complex learning and inference procedures. Thus, as the size of knowledge graphs has grown, recent work has focused more on *knowledge graph embedding* models for relational learning as an alternative to the traditional symbolic approaches. These models embed entities and relations in a low-dimensional continuous latent space, while trying to preserve the information present in the graph.

Although latent feature models have been successful in scaling to larger knowledge graphs and achieving higher scores on various metrics, the kind of information captured by these embeddings is still not fully understood. Analyzing these embeddings may allow us to better understand the strengths and weaknesses of these models. A common drawback of these models is their inability to model all of the basic relational patterns in real-world knowledge graphs. Different models are implicitly capable of learning specific relational patterns. For

example, DistMult [47] can learn symmetric and transitive patterns, while TransE [6] is well suited for inversion and transitive patterns. In practice however, these methods often find it difficult to learn these patterns, either due to insufficient data to generalize well, or due to the complexity of relational data in the larger real-world knowledge graphs. Whereas simple rule-based methods can extract these patterns and provide much better performance for small and reasonably sized knowledge graphs [42].

In this work, we investigate the logical inference capabilities of latent embedding models. Specifically, we empirically validate the suitability of some of the most commonly used embedding models for learning different kinds of relational patterns. We study the relationship between embedding dimensionality, knowledge graph size (number of entities), and the link prediction performance of the models. To perform this analysis, for each type of pattern we generate several synthetic datasets of varying sizes. This allows us to study the impact of different scoring functions used by these models on the performance on each pattern type specifically. By varying the dimensionality of latent space, we can observe the changes in the ability of the models to learn the observed data (relational patterns) under different constraints.

Chapter 2

Literature Review

Knowledge graphs store information as a graph of objects (nodes) and relationships (edges) between these objects. With the growing number of such knowledge graphs and their applications in numerous domains, the knowledge graph completion problem has received widespread attention in recent years. As the size of these graphs has grown exponentially, the research focus has shifted towards more scalable relational learning models.

2.1 Notations

Let $\mathcal{E}$ be the set of entities, and $\mathcal{R}$ be the set of relations stored in the graph. The set of all possible triplets can be defined as $\mathcal{T} = \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. Let knowledge graph be represented as a collection of triplets, KG = $\{h, \mathrm{r}, t)\}$, where h, $t \in \mathcal{E}$ and $\mathrm{r} \in \mathcal{R}$. We use small letters to describe entities and relations, and respective bold letters to describe their vectors representations. A relation in matrix form is represented by a capital letter M_r, with dimensions $d_r \times d_e$.

2.2 Observed Feature Models

The earlier techniques focused on symbolic representations of entities and relations using first-order or propositional logic, and used probabilistic graphical models for learning and inference. These techniques can be categorized under the field of Statistical Relational Learning, also known as Relational Machine Learning. Some of the most applied methods probabilistic approaches from SRL are the Markov Logic Networks (MLN) [35], Relational Dependency Networks (RDN) [31], Bayesian Logic Programs (BLP) [20] and Probabilistic Relational Models (PRM) [21]. Other methods, such as AMIE [14] and ALEPH [39] extract rules from the relational data and then use these rules to infer new information. AMIE uses frequency of rule instantiation (support) in knowledge graph while ALEPH uses Inductive Logic Programming [29] for extracting the rules.

The graph-based feature models use characteristics of the knowledge graph, such as neighboring entities and multi-relational paths between entities as features for link prediction. Perhaps the most used graph-based feature model is the Path Ranking Algorithm (PRA) [22, 23]. The PRA performs fixed-length random walks to discover a set of relation paths specific for each relation, which are useful for predicting that particular relation. For each triplet (h, r, t), it then determines the probabilities of existence of these relation specific paths between the entity pair (h, t). Using these path probabilities as features corresponding to a pair of entities, the algorithm can then use any classifier, such as logistic regression, to compute the score (probability) of the triplet. The step of computing path probabilities between entity pairs is computational expensive. Thus, instead of computing relation path probabilities, [30] used a bigger set of features, but constraints the features to have binary values. [15] show that this step is not beneficial, and instead propose a new PRA-based model called Subgraph Feature Extraction (SFE). The SFE model constructs a neighborhood for each node using random walks, and uses this information to generate binary feature vectors for entity pairs. These features are much more expressive than the relation paths

used in PRA, allowing the feature vectors to capture more information with much lower computational cost. Distinct Subgraph Paths (DSP) [28] further improves upon SFE. The model uses a new set of features that describe distinct properties of entities using disjoint sets of subgraph paths for both subject and object entities.

Both the SRL models and the graph feature models provide easily interpretable results, as the new edges can be described in term of the existing edges and/or probabilistic weighed relational rules derived from the knowledge graph.

2.3 Latent Feature Models

The latent embedding methods learn representations of relations and entities in a continuous latent space. The scoring function $\varphi : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \rightarrow \mathbb{R}$ defined as a function of entity and relation embeddings computes the probability of a triplet (h, r, t). These models can scale almost linearly with the number of triplets, entities and relations. The knowledge graph embeddings methods can be described using three properties: the representation of entities and relations, the scoring function for computing the score/probability, and a loss function to train the model. The various functions described below differ primarily on the first two properties. All the methods described here rely on local closed-world assumption. The key idea is to assign a higher score to triplets observed in KG than to triplets that were not observed. Based on the scoring function, these methods can be roughly categorized into two categories: translational latent models and semantic-based latent models.

2.3.1 Translational latent models

The translational latent models or latent distance models use a scoring function which measures the distance between the translated embeddings of entities to compute the probability

of a triplet. The translation is usually unique for each relation. The triplets where the transformed entity embeddings are closer to each other receive higher scores. The earliest model of this kind is the Structured Embeddings (SE) [7] model. It computes the score of a triplet as:

$$\varphi(h, r, t) = - \parallel M_r^s \mathbf{h} - M_r^o \mathbf{t} \parallel_1$$

Here, the matrices transform the head and tail entity embeddings differently for each relation. TransE [6] reduces the number of parameters by removing the matrix transformations. It represents the relations and entities as vectors, and offsets the head entity embedding by the relation embedding:

$$\varphi(h, r, t) = - \parallel \mathbf{h} + \mathbf{r} - \mathbf{t} \parallel$$

As can be observed from the scoring function, the model struggles with 1-N and N-1 mapping functions. For instance, for N-1 relations, all head entities need to have very similar embeddings in order to be close to the tail entity after relational translation. To deal with these drawbacks, other modes use relation specific embeddings for entities, so that entities in 1-N and N-1 relations can have very similar embeddings for one relation, but completely different embeddings for other relations. Unstructured Model [3] is a simplified version of TransE, which cannot differentiate between relations:

$$\varphi(h, r, t) = - \parallel \mathbf{h} - \mathbf{t} \parallel_2^2$$

TransH [45] computes the distance similar to TransE, but considers the projections of head and entity embeddings onto a relation-specific hyperplane:

$$\mathbf{h}' = \mathbf{h} - \boldsymbol{w}_r^T \mathbf{h} \boldsymbol{w}_r \qquad \mathbf{t}' = \mathbf{t} - \boldsymbol{w}_r^T \mathbf{t} \boldsymbol{w}_r$$

$$\varphi(h, r, t) = - \parallel \mathbf{h}' + \mathbf{r} - \mathbf{t}' \parallel$$

TransR [25] represents entities in entity spaces $\mathbb{R}^k$ and relations in relation spaces $\mathbb{R}^d$. The entity embeddings is projected to relation space using a relation specific matrix M_r:

$$\mathbf{h}' = M_r\mathbf{h} \qquad \mathbf{t}' = M_r\mathbf{t}$$

$$\varphi(h, r, t) = - \parallel \mathbf{h}' + \mathbf{r} - \mathbf{t}' \parallel$$

Other models in this category are TransA [46], TransM [12], TransF [13], TransD [18], etc.

2.3.2 Semantic latent models

The semantic latent models compare the similarity between the entity embeddings in latent space. The semantic matching methods use similarity-based loss functions which rely on matching the embeddings of entities and relation for computing the probability of a triplet.

RESCAL [34] represents entities as vectors and relations as matrices. It captures pairwise interactions between the head and tail entities via a relation matrix:

$$\varphi(h, r, t) = \boldsymbol{h}^T M_r \mathbf{t}$$

DistMult [47] simplifies the RESCAL model by restricting the relation matrix to be a diagonal matrix. However, it cannot deal with asymmetric relations. ComplEx [43] is an extension of DistMult to complex space. It represents entities as complex-valued vectors, and relations as complex-valued diagonal matrices. It captures pairwise interaction between head entity embedding and complex conjugate of tail entity embedding. The score of a triplet can be computed as:

$$\varphi(h, r, t) = Re\left(\mathbf{h}^T M_r \bar{\mathbf{t}}\right)$$

The RESCAL and similar models attempt to capture all pairwise interactions between entities. This can result in a large number of parameters, if the number of relations is large. The semantic latent models based on neural networks provide a more scalable alternative, by learning only the useful interactions, instead of considering all possible interactions. The Semantic Matching Energy (SME) [5] combines the relation vector with head and tail entity vectors separately, and finally uses a dot product to compute the score of the triplet:

$$\varphi(h, r, t) = g_u(\mathbf{h}, \mathbf{r})^T \; g_u(\mathbf{t}, \mathbf{r})$$

$$g_u(\boldsymbol{h}, \boldsymbol{r}) = M_u^1 \mathbf{h} + M_u^2 \mathbf{r} + \mathbf{b}_u \qquad g_v(\boldsymbol{t}, \boldsymbol{r}) = M_v^1 \mathbf{t} + M_v^2 \mathbf{r} + \mathbf{b}_v$$

Similarly, E-MLP [38] also uses neural network to combine entity embeddings using a relation-specific transformation matrix:

$$\mathbf{h}_{htr}^a = M_r^T [\mathbf{h}; \mathbf{t}]$$

$$\varphi(h, r, t) = \boldsymbol{w}_r^T g(\mathbf{h}_{htr}^a)$$

ER-MLP [10] represents relation as a vector and feeds it into the neural network along with the entity embeddings. The transformation is independent of the relation, hence the number of parameters is substantially reduced:

$$\mathbf{h}_{htr}^a = \boldsymbol{C}^T [\mathbf{h}; \mathbf{r}; \mathbf{t}]$$

$$\varphi(h, r, t) = \boldsymbol{w}^T g(\mathbf{h}_{htr}^a)$$

Neural Tensor Network (NTN) [38] is a generalization of RESCAL, combining neural network based and bilinear models. The NTN model however has much more parameters than both RESCAL and ER-MLP, and tends to overfit. RotatE [41] model describes each relation as

a rotation from head entity to the tail entity in the complex space. The scoring function is an element-wise difference between the rotated head entity and the tail entity:

$$\varphi(h, r, t) = - \parallel \mathbf{h} \circ \mathbf{r} - \mathbf{t} \parallel$$

Here, $\circ$ denotes the element-wise product. ConvE [9] uses convolutional neural network to compute the interactions between entity and relation embeddings. The entities and relations are represented as vectors. The score is computed as:

$$\varphi(h, r, t) = f(vec(f([\mathbf{h}; \mathbf{r}] * w\,)))\,\mathbf{t}$$

Here, f is a non-linear function, usually the rectifier linear unit, and w is the convolutional filter. R-GCN [37] is an extension of the GCN [11] to multi-relational data. The R-GCN model is similar to an auto-encoder, where the encoder part generates vector representations for entities and relations, and the decoder part uses these embeddings to compute the triplet scores. To compute the representations for entities, R-GCN combines information from local graph neighborhood. The embedding $\mathbf{h}_i^{(l+1)}$ is calculated as:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{r \in R} \sum_{j \in N_i^r} \frac{1}{c_{i,r}} W_r^{(l)} \mathbf{h}_j^{(l)} + W_0^{(l)} \mathbf{h}_j^{(l)}\right)$$

This can be viewed as a normalized sum of transformed neighborhood entities. N_i^r represents the neighboring entities of node i under the relation r, and $c_{i,r}$ is a constant that can be either chosen in advance or learnt during training. Each relation has its own transformation matrix W_r. The model uses DistMult method as the decoder/scoring function.

2.4 Hybrid models

As observed in [10, 42], the benefits of graph feature models and latent feature models are complementary. Latent feature models work better for large datasets with simple relational patterns. On the other hand, graph feature models work better for smaller datasets with more complex patterns. Thus, several models attempt to combine the two approaches to improve performance. The combined model is often times much faster to train. Additive Relational Effects (ARE) [32] combines RESCAL with Path Ranking Algorithm:

$$\varphi(h, r, t) = \mathbf{w}_r^{(1)T} \boldsymbol{\phi}_{ht}^{RESCAL} + \mathbf{w}_r^{(2)T} \boldsymbol{\phi}_{hrt}^{PRA}$$

The model is trained by alternately optimizing both the models, and allows for much lower-dimensionality for RESCAL. [19] combines latent features with additional information about existence of other edges between two entities to predict a new edge. Some approaches use auxiliary information such as entity types and textual mentions to supplement the information present in knowledge graphs. [24] augments existing knowledge graph with syntactic relational information from a web text corpus. [16] builds upon this model by using relations with lexicalized syntactic labels to augment the KG instead of unlexicalized dependency role labels. [42] uses a simple observed feature model and outperforms the state-of-the-art latent feature models on FB15K [6] and WN18 [6] datasets. It also evaluates the models on a new dataset by removing redundant relations and augmenting with textual mentions from a document collection. The combination of observed and latent feature models performs better than either of them individually. The learning method is similar to [24] and [16], but modifies the loss function to focus more on learning the knowledge graph relations.

An alternate approach combines first-order logic and latent feature models by injecting information from prior logical rules into the embeddings. [44] represents training as an Integer Linear Programming problem, and uses logical rules to device the constraints. KALE

[17] models knowledge and logic jointly to learn entity and relation embeddings. The model uses t-norm fuzzy logics to represent knowledge graph triplets as well as logic rules in a unified framework. [36] uses matrix factorization and first-order logic to learn low-dimensional *logic embeddings* for entity-pairs and relations.

Chapter 3

Experiments

The existing real-world knowledge graphs comprise of different kinds of relation patterns. These patterns influence the performance of various knowledge graph link prediction models, as different models are implicitly suited to capture different such patterns. We synthesize multiple datasets, each corresponding to a specific relational pattern, and evaluate several of the existing models on these datasets. This comparative analysis helps us determine the suitability of models for learning different patterns, and identify some limitations of these models.

3.1 Data sets

To evaluate the knowledge graph embedding models on different relational patterns, we generate multiple synthetic datasets. We also experiment with several kinds of embedding based models, including the state-of-the-art RotatE [41] model.

Data set	Number of Entities	Number of Relations	Number of Triplets		
			Training	Validation	Test
FB15k-237	14,541	237	272,115	17,535	20,466
WN18RR	40,943	11	86,835	3,034	3,134

Table 3.1: Data sets statistics.

3.1.1 Real world Data sets: FB15k-237 and WN18RR

This section describes two of the most popular benchmark knowledge graphs used for evaluating link prediction models:

- FB15k-237 [42]: Freebase[2] is a large-scale knowledge graph containing facts about general human knowledge, with more than a billion triplets and around 100 million entities. [6] presented a smaller subset of Freebase, removing entities and relations not present in Wikilinks database and having at least 100 mentions in Freebase. The newer smaller data set has 592,213 triplets with 14,951 entities and 1,345 relations, and is referred to as FB15k. [42] observed that almost 81% of test triplets (x, r, y) could be inferred directly via a directly linked triplet (x, r', y) or (y, r', x). Thus, a new dataset FB15k-237 was proposed where inverse relations have been removed. This dataset consists of mainly transitive relational patterns.

- WN18RR [9]: WordNet [27] is a huge database of English, consisting of lexical relations. It contains relations between nouns, verbs, adjectives and adverbs. The subset of WordNet, WN18 [4] also suffers from test leakage like FB15k, and thus WN18RR was proposed. This dataset consists of predominantly symmetric relations.

The statistics for aforementioned knowledge graphs are provided in Table 3.1

Relation pattern	Relational clausal form
Symmetric	x R y $\Rightarrow$ y R x
Equivalence	x R_1 y $\Rightarrow$ x R_2 y
Inversion	x R_1 y $\Rightarrow$ y R_2 x
Transitive (single relation)	x R y $\wedge$ y R z $\Rightarrow$ x R z
Transitive (multiple relations)	x R_1 y $\wedge$ y R_2 z $\Rightarrow$ x R_3 z

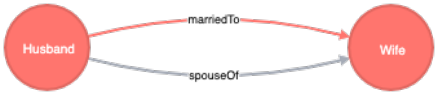
Table 3.2: Relational pattern types.

3.1.2 Synthetic Data sets

To test our hypothesis, we experiment with synthetic datasets of different sizes. The existing real-world knowledge graphs comprise of different relational patterns. These graphs are usually dominated by certain kinds of patterns. This provides an intuition as to which models are more suitable for these graphs, based on different models' suitability for learning certain types of patterns. For instance, FB15k-237 contains predominantly composition relations. Thus, TransE performs well as it can capture the transitive relational patterns. However, as the real-world KGs have a complex mix of different types of relations and relational patterns, this performance variation simply between models cannot be used to conclude about their capabilities in learning a specific pattern.

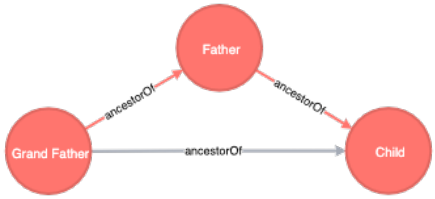
We generate datasets for five different pattern types, as summarized in Table 3.2. For each pattern, we generate datasets of four different sizes (in terms of number of triplets/entities). We generate training sets of sizes $\in$ {1000, 5000, 10000, 20000} by adding multiple sets of relational clause instantiations until the required number of triplets is obtained. For generating validation/test sets, we generate 600 sets of triplets for each pattern. The triplet corresponding to the head of the pattern clause is added to the test set, while the triplets corresponding to the body of the clause are added to the training set (Figure 3.1).

(a) Symmetric

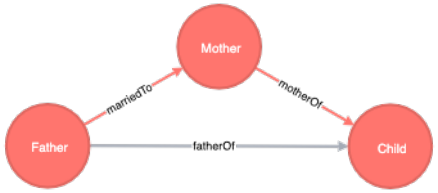


(b) Equivalence

(c) Inverse



(d) Transitive with same relation



(e) Transitive with different relations

Figure 3.1: Examples of different relational patterns. The triplet corresponding to the head (black edge) is to be predicted and added to the test set, while the triplets corresponding to the body of the clause (red edges) are added to training set.

3.2 Evaluation Metrics

We evaluate the models using two standard metrics for link prediction, Mean Reciprocal Rank (MRR) and Hits@n. For each triplet in the evaluation set, we rank the triplet along with all the candidate triplets in descending order of their probabilities/scores. The candidate triplets are computed by corrupting either the head entity or the tail entity. MRR denotes the average of the inverse rank of all the triplets in the evaluation set. Before ranking, we remove the triplets which exist in the training, validation or test set, to compare the triplet only with false candidate triplets. This setting is referred to as the *filtered setting* [6].

3.3 Implementation

For ConvE, we use the implementation provided by [9]. For remaining models, we use implementation provided by [41]. For each relational pattern, we train all the models with different embedding dimensionality d $\in \{10, 25, 50, 75, 100, 200, 400\}$. We add an additional early stopping condition for regularization, terminating when the MRR on validation set does not improve for three consecutive validation steps. For training each model, we choose their respective hyper-parameters tuned on the FB15k-237 dataset. Both the models use negative sampling for training.

The ConvE implementation uses binary cross-entropy loss function:

$$\mathcal{L}(h, r) = -\frac{1}{N} \sum_{i} (y_i \cdot log(p_i) + (1 - y_i) \cdot log(1 - p_i))$$

Here, we compute the scores for all possible tail entities simultaneously (1-N scoring). Thus, $p_i = \sigma(\varphi(h, r, t_i))$ and y_i is the true label for the triplet (h, r, t_i). The RotatE implementation uses margin-based loss function:

$$\mathcal{L}(h, r, t) = -log\ \sigma(\gamma - \varphi(h, r, t)) - \sum_{i}^{n} \frac{1}{k}\ log\ \sigma(\varphi(h'_i, r, t'_i) - \gamma)$$

Here, γ is the margin hyper-parameter, and (h'_i, r, t'_i) is the i^{th} negative triplet.

Chapter 4

Results and Analysis

4.1 Link Prediction for Relational Patterns

In this section, we report the performance of knowledge embedding models on link prediction task for various relational patterns. The results are summarized in Table 4.1.

4.1.1 Symmetric Pattern

The models perform very well on the symmetry pattern between two relations (Figure 4.1). TransE model fails to learn because the symmetry pattern forces the relation embeddings to be close to $\mathbf{0}$. For lower embedding dimensions (less than equals 25), RotatE and ComplEx models perform far better than ConvE and DistMult, which shows their superiority in learning this specific pattern. For lower dimensions, performance falls slightly as size increases. In case of dimensionality 10, the performance falls to zero for all models. Thus, all models other than TransE can capture this pattern.

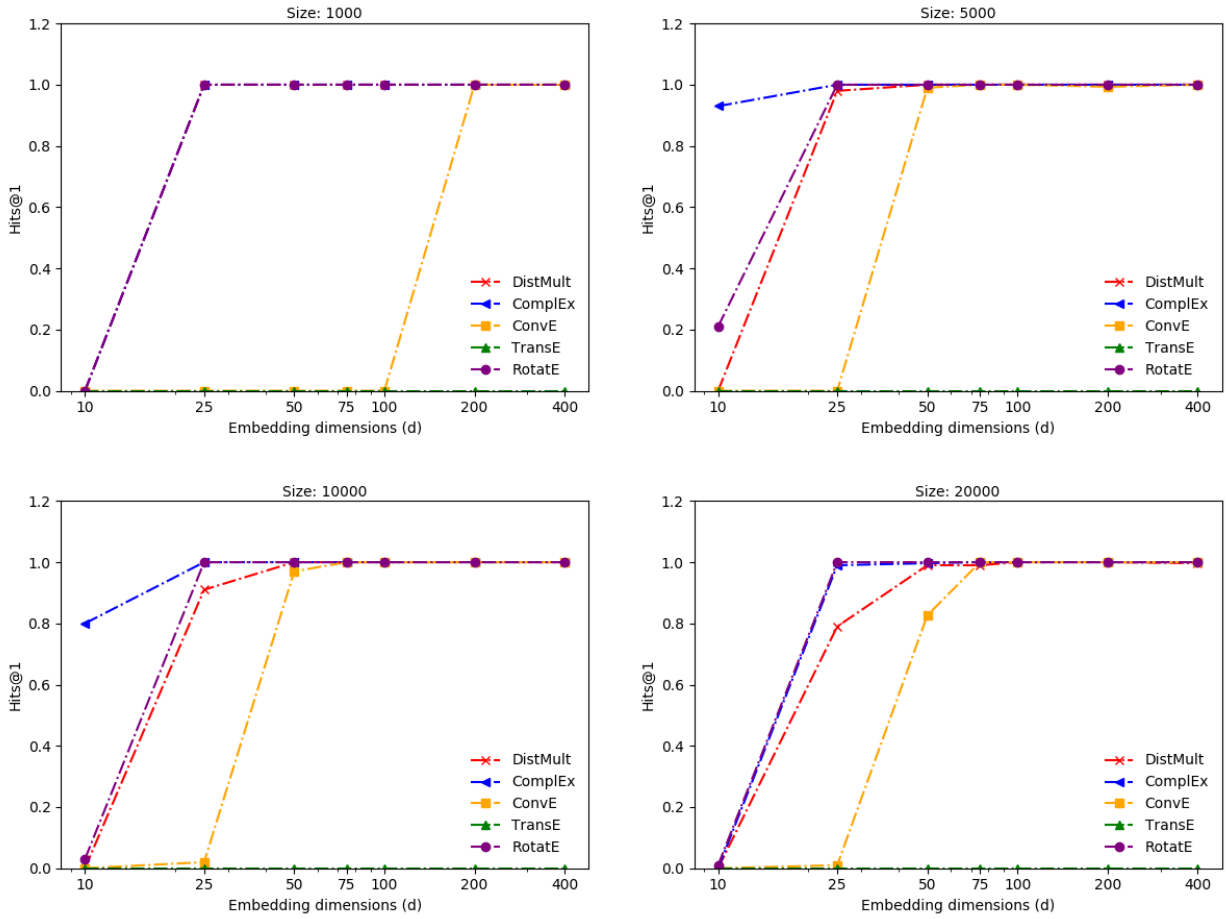


Figure 4.1: Performance vs Embedding dimensions for *symmetric* relational pattern.

4.1.2 Equivalence Pattern

The performance trend on equivalence pattern is very similar to the symmetric pattern (Figure 4.2). This pattern is presumably the easiest to learn, as all the models can effectively learn near-identical embeddings for both the relations. All the models are able to capture equivalence between two different relations. Similar to symmetric pattern, the model performance drops slightly as size increases for low dimensions. RotatE performs best for very low dimensionality (10).

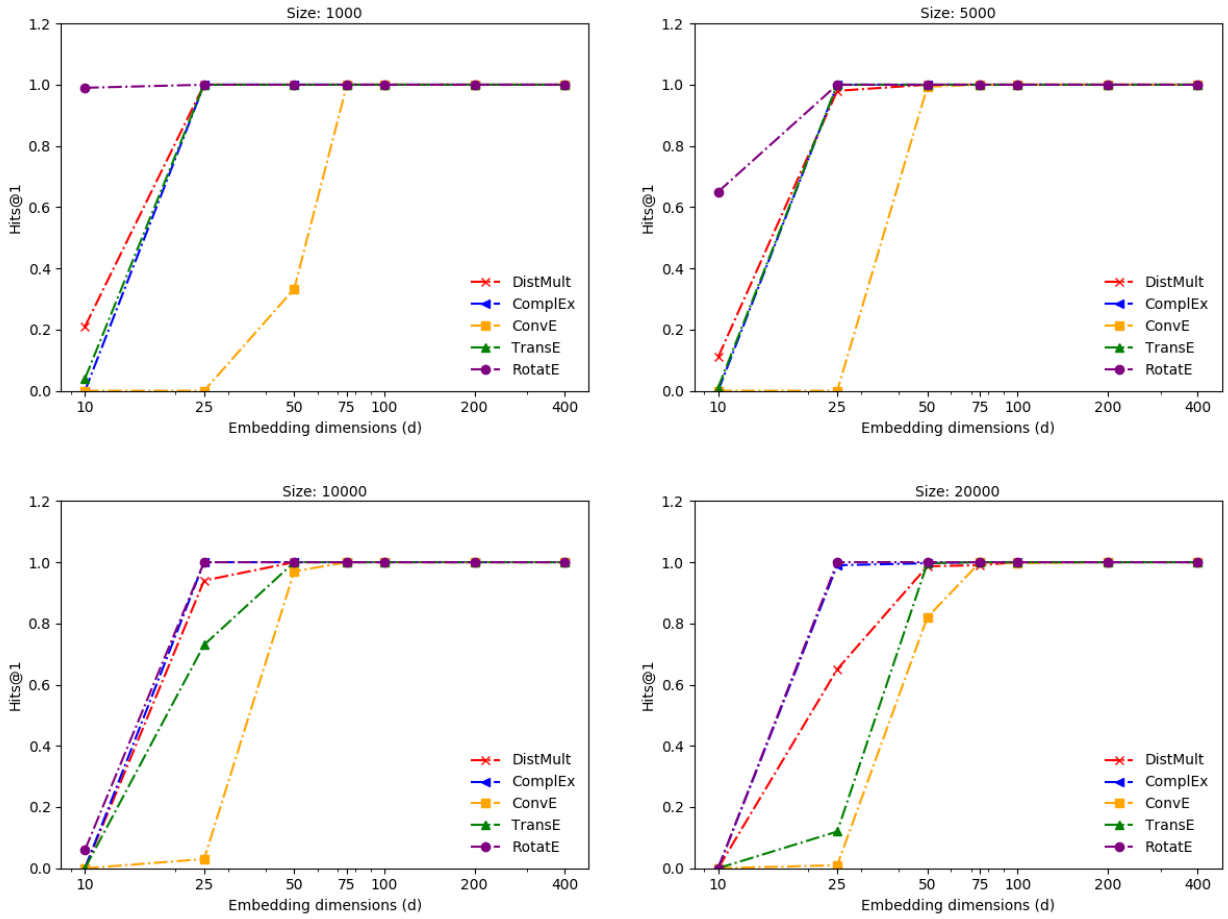


Figure 4.2: Performance vs Embedding dimensions for *equivalence* relational pattern.

4.1.3 Inversion pattern

The performance trend on equivalence pattern is almost identical to the equivalence pattern (Figure 4.3). All the models are able to capture inversion between two relations. The minimum embedding dimension for models to learn perfectly increases from 25 to 75 as the dataset size increases. RotatE again outperforms other models for very low dimensionality (10) on mid-sized datasets, but performance drops back to 0 for bigger datasets.

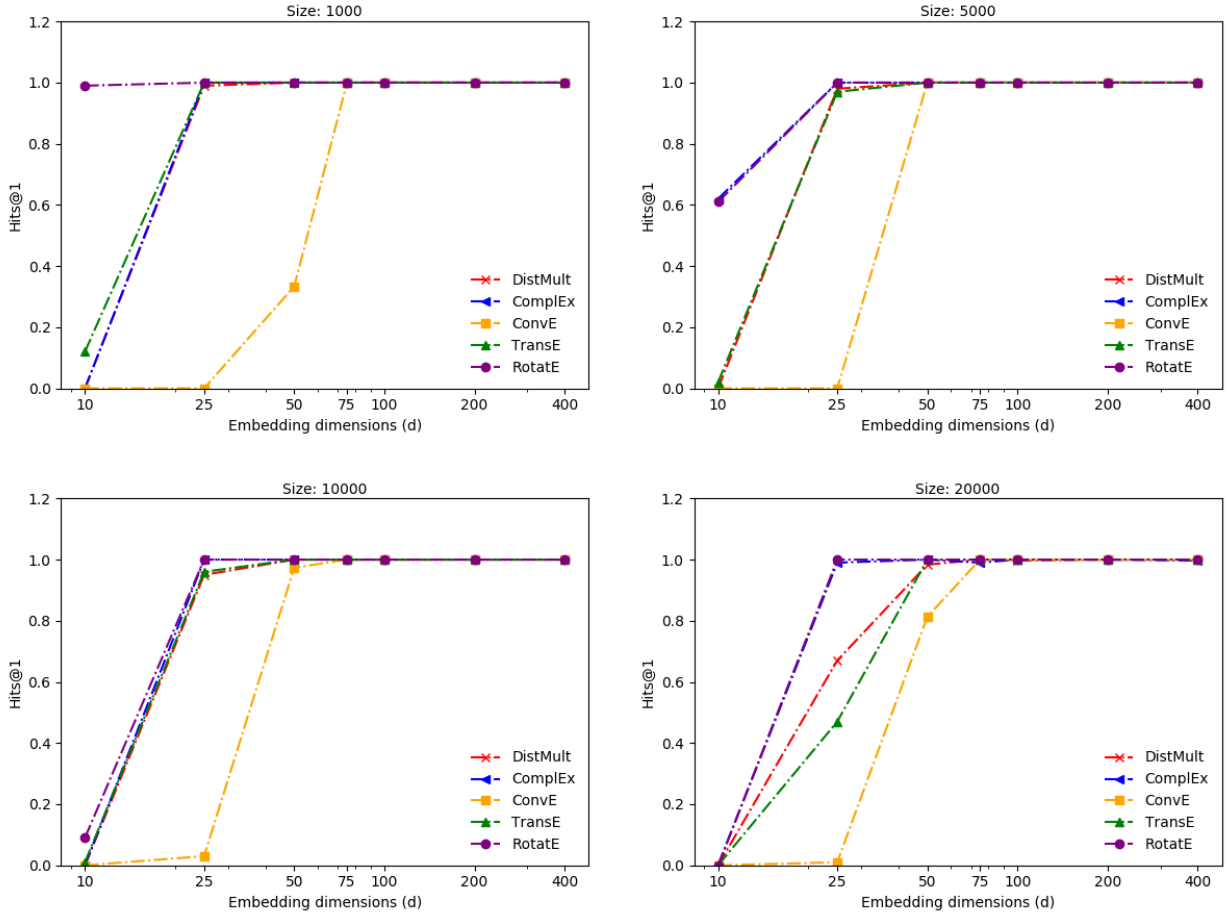


Figure 4.3: Performance vs Embedding dimensions for *inversion* relational pattern.

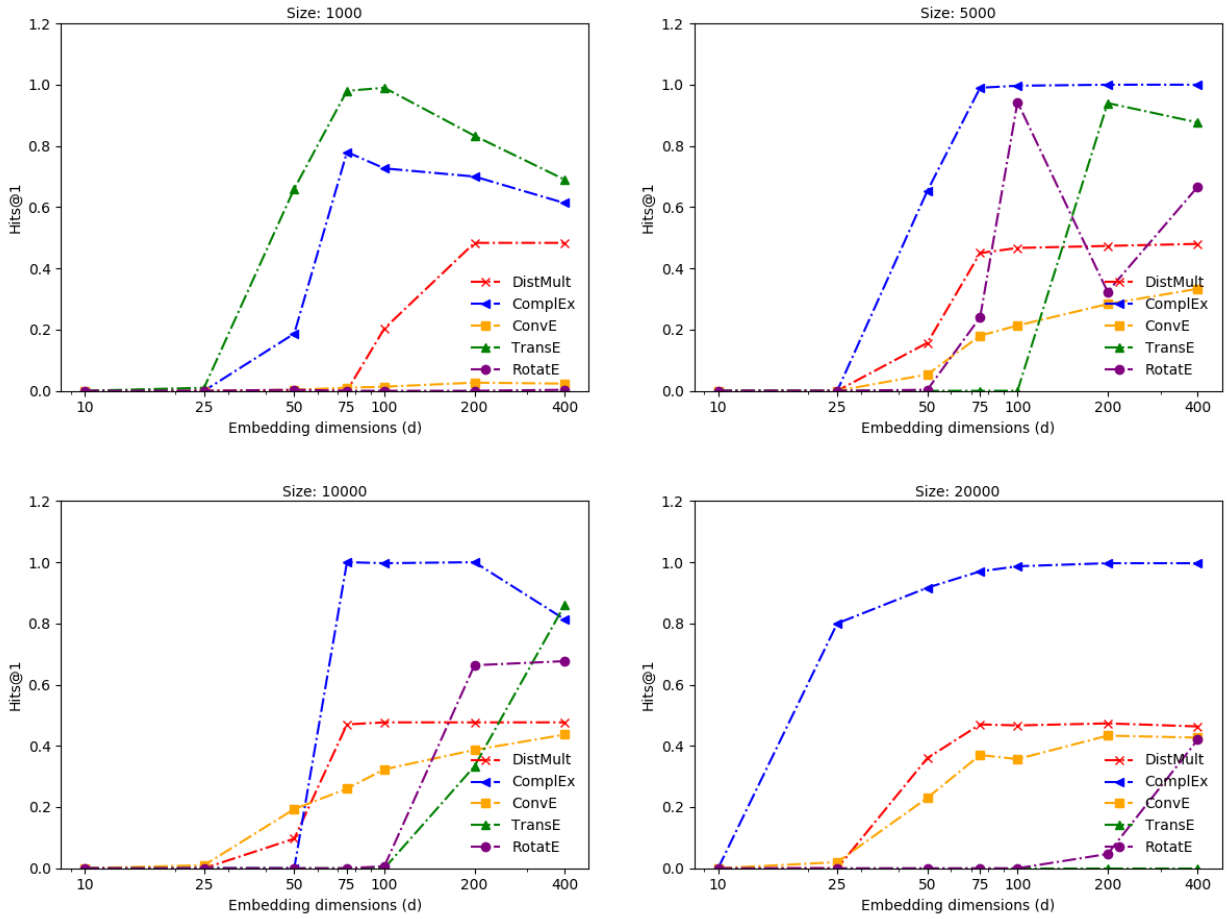


Figure 4.4: Performance vs Embedding dimensions for *transitive* relational pattern with same relation.

4.1.4 Transitive pattern with same relation

This pattern is the most difficult to learn for models. The models perform poorly for low dimensionality (less than equals 25). The performance increases after 25, and saturates for higher dimensions. TransE performs very well for small dataset size, but the performance degrades rapidly with increasing size, resulting in zero Hits@1 score for biggest dataset size irrespective of dimensionality. DistMult and ConvE perform better for higher dimensions, but still fail to learn this pattern. RotatE also performs poorly for this pattern, especially on bigger datasets. ComplEx performs poorly for low dimensions and small datasets, but outperforms all models on bigger datasets.

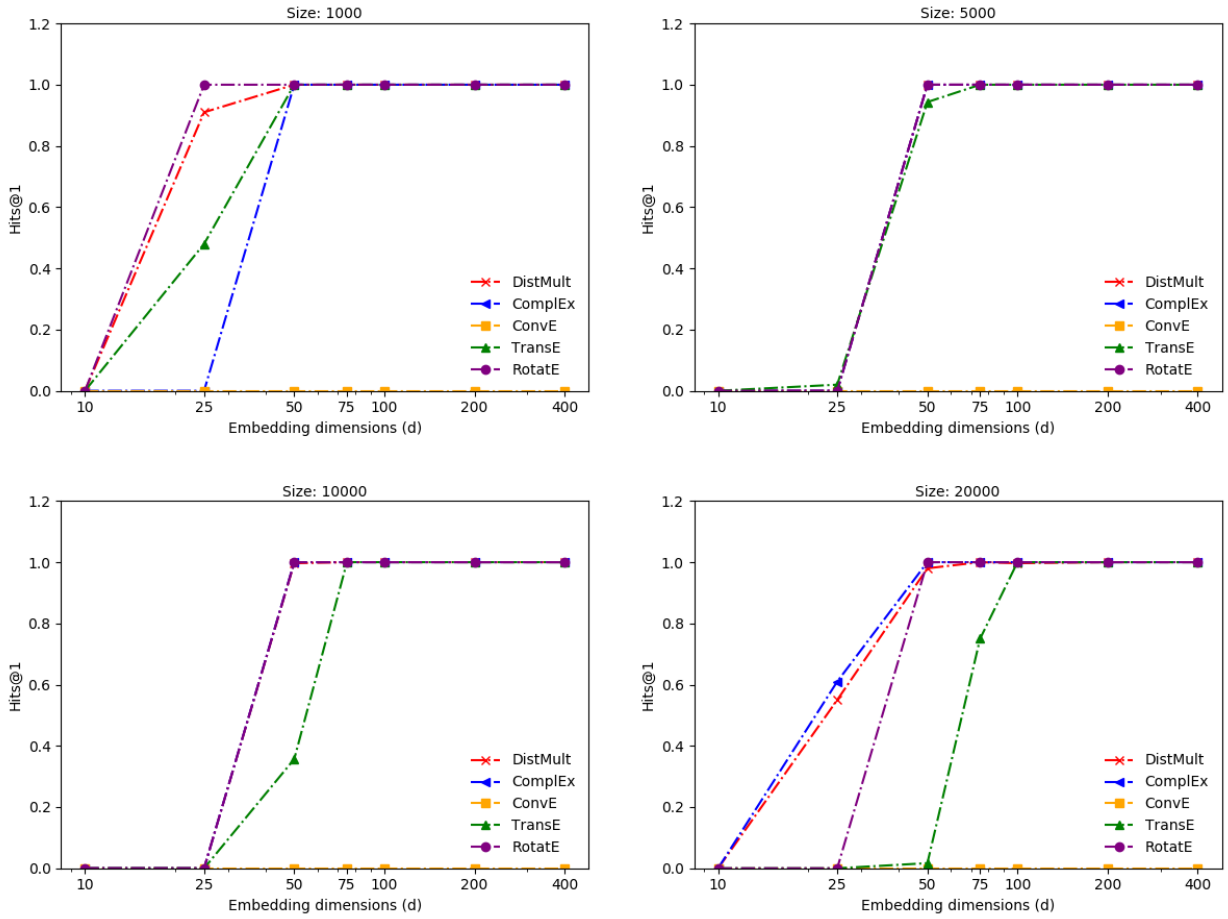


Figure 4.5: Performance vs Embedding dimensions for *transitive* relational pattern with different relations.

4.1.5 Transitive pattern with different relations

The models perform much better for transitivity involving different relations than with the same relation. ConvE completely fails to learn the pattern, irrespective of dataset size and dimensionality. For lower dimensions, the performance drops slightly as dataset size increases. For ComplEx and DistMult, however, the performance improves for dimensionality 25. ComplEx performs best for lower dimensionality, and all the models other than ConvE achieve perfect score for dimensionality higher than or equal to 100. Thus, all the models other than ConvE can learn this pattern.

Pattern	TransE	DistMult	ComplEx	RotatE	ConvE
Symmetric	✗	✔	✔	✔	✔
Equivalence	✔	✔	✔	✔	✔
Inversion	✔	✔✔	✔	✔	✔
Transitive (same relation)	✗	✗	✔	✗	✗
Transitive (different relations)	✔	✔✔	✔✔	✔	✗

Table 4.1: Logical inference capability for different relational patterns. Double tick marks contradicting results.

4.2 Qualitative Analysis

We observe that different models perform better for different data patterns. Some of these results are contradicting to existing work. For instance, DistMult is not expected to learn inversion pattern, but successfully models the synthetic dataset for inversion pattern. Similarly, ComplEx outperforms other models on transitive pattern. This difference is more easily observed for lower dimensions, but as dimensionality increases, all models can effectively learn the simple logical patterns. This is not applicable in some instances, such as the symmetry pattern for TransE, or the anti-symmetry pattern for DistMult, due to the intrinsic nature of their scoring functions. As expected, the minimum dimensionality required to learn the patterns is higher for transitive patterns compared to symmetric/inversion patterns. Thus, more complex patterns require more dimensionality for the same dataset size. The performance of these models starts to decrease as the dataset size increases. Thus, as the dataset size increases, the models require more dimensionality to be able to model the data. This suggests that these models can possibly capture simple relational patterns present in large real-world knowledge graphs as well, if provided with sufficiently large embedding dimensionality.

Chapter 5

Conclusion

In this work, we explored the logical inference capabilities of latent embedding models. We evaluate the link prediction capabilities of existing knowledge graph embedding models on different kinds of relational patterns. For unbiased evaluation, we generate synthetic datasets of different sizes for each pattern, comprising of triplets encoding that specific pattern. We observe that the models are able to learn most of the patterns achieving high scores on both metrics, provided sufficiently large embeddings. For lower dimensionality, however, the performance varies considerably between models. It suggests that in most cases, these latent models can capture the simple logical patterns when provided with sufficient representation power. Due to difficulty in interpretability of the embeddings learnt, the reason for performance variation of different models on real-world datasets is still not completely understood. It maybe due to presence of much more complex relational patterns which cannot be captured by these models, due to lower embedding dimensionality, or due to some other intrinsic features of these datasets. In future work, we would like to evaluate models on more complex relational patterns and with higher dimensionality. Analyzing the real-world knowledge graphs for such complex patterns may also provide some key insights into our understanding of these graphs as well as these embedding models.

References

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[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 SIGMOD international conference on Management of data*, pages 1247–1250. AcM, 2008.

[3] A. Bordes, X. Glorot, J. Weston, and Y. Bengio. Joint learning of words and meaning representations for open-text semantic parsing. In *Artificial Intelligence and Statistics*, pages 127–135, 2012.

[4] A. Bordes, X. Glorot, J. Weston, and Y. Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259, 2014.

[5] A. Bordes, X. Glorot, J. Weston, and Y. Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259, 2014.

[6] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc., 2013.

[7] A. Bordes, J. Weston, R. Collobert, and Y. Bengio. Learning structured embeddings of knowledge bases. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[8] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[9] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[10] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM, 2014.

[11] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[12] M. Fan, Q. Zhou, E. Chang, and T. F. Zheng. Transition-based knowledge graph embedding with relational mapping properties. In *Proceedings of the 28th Pacific Asia Conference on Language, Information and Computing*, 2014.

[13] J. Feng, M. Huang, M. Wang, M. Zhou, Y. Hao, and X. Zhu. Knowledge graph embedding by flexible translation. In *Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2016.

[14] 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*, pages 413–422. ACM, 2013.

[15] M. Gardner and T. Mitchell. Efficient and expressive knowledge base completion using subgraph feature extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1488–1498, 2015.

[16] M. Gardner, P. P. Talukdar, B. Kisiel, and T. Mitchell. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 833–838, 2013.

[17] S. Guo, Q. Wang, L. Wang, B. Wang, and L. Guo. Jointly embedding knowledge graphs and logical rules. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 192–202, 2016.

[18] 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 (Volume 1: Long Papers)*, volume 1, pages 687–696, 2015.

[19] X. Jiang, V. Tresp, Y. Huang, and M. Nickel. Link prediction in multi-relational graphs using additive models. *SeRSy*, 919:1–12, 2012.

[20] K. Kersting and L. De Raedt. Bayesian logic programs. *arXiv preprint cs/0111058*, 2001.

[21] D. Koller. Probabilistic relational models. In *International Conference on Inductive Logic Programming*, pages 3–13. Springer, 1999.

[22] N. Lao and W. W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67, 2010.

[23] N. Lao, T. Mitchell, and W. W. Cohen. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 529–539. Association for Computational Linguistics, 2011.

[24] N. Lao, A. Subramanya, F. Pereira, and W. W. Cohen. Reading the web with learned syntactic-semantic inference rules. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1017–1026. Association for Computational Linguistics, 2012.

[25] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[26] H. Liu and P. Singh. Conceptnet?a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226, 2004.

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

[28] S. K. Mohamed, V. Nováček, and P.-Y. Vandenbussche. Knowledge base completion using distinct subgraph paths. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, pages 1992–1999. ACM, 2018.

[29] S. Muggleton. Inductive logic programming. *New generation computing*, 8(4):295–318, 1991.

[30] A. Neelakantan, B. Roth, and A. McCallum. Compositional vector space models for knowledge base inference. In *2015 aaai spring symposium series*, 2015.

[31] J. Neville and D. Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 8(Mar):653–692, 2007.

[32] M. Nickel, X. Jiang, and V. Tresp. Reducing the rank in relational factorization models by including observable patterns. In *Advances in Neural Information Processing Systems*, pages 1179–1187, 2014.

[33] M. Nickel, L. Rosasco, and T. Poggio. Holographic embeddings of knowledge graphs. In *Thirtieth Aaai conference on artificial intelligence*, 2016.

[34] M. Nickel, V. Tresp, and H.-P. Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, volume 11, pages 809–816, 2011.

[35] M. Richardson and P. Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.

[36] T. Rocktäschel, S. Singh, and S. Riedel. Injecting logical background knowledge into embeddings for relation extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1119–1129, 2015.

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

[38] R. Socher, D. Chen, C. D. Manning, and A. Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in neural information processing systems*, pages 926–934, 2013.

[39] A. Srinivasan. The aleph manual, 2001.

[40] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

[41] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

[42] 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*, pages 57–66, 2015.

[43] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080, 2016.

[44] Q. Wang, B. Wang, and L. Guo. Knowledge base completion using embeddings and rules. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[45] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI conference on artificial intelligence*, 2014.

[46] H. Xiao, M. Huang, Y. Hao, and X. Zhu. Transa: An adaptive approach for knowledge graph embedding. *arXiv preprint arXiv:1509.05490*, 2015.

[47] B. Yang, W.-t. Yih, X. He, J. Gao, and L. Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

Appendices

A Evaluation Metrics

The standard metrics for evaluating the performance of link prediction methods are the Mean Reciprocal Rank (MRR) and Hits@n. The objective of link prediction is to predict tail entity given subject entity and relation, or subject entity given tail entity and relation. The key idea is that triplets occurring in the KG should be scored higher than the corresponding false corrupted triplets. For each triplet in the test set $\mathcal{T} = \{ (h, r, t) \}$, we corrupt each triplet by replacing the tail (or head) entity with all the remaining entities $e' \in \mathcal{E}$ and rank all the triplets based on the probabilities of (h, r, e') (respectively (e', r, t)). As some of these corrupt triplets might be true, we remove any corrupted triplet from the candidate set that exists in the graph (including validation/test sets). This is referred to as the *filtered* setting, introduced by [6].

We define the left and right rank of a triplet (h, r, t) depending on whether we corrupt the head entity or the tail entity:

$$rank_i^{left} = 1 + \sum_{(h', r_i, t_i) \notin \mathcal{T}} \boldsymbol{I} \left[\varphi(h', r_i, t_i) > \varphi(h_i, r_i, t_i) \right]$$

$$rank_i^{right} = 1 + \sum_{(h_i, r_i, t') \notin \mathcal{T}} \boldsymbol{I} \left[\varphi(h_i, r_i, t') > \varphi(h_i, r_i, t_i) \right]$$

Here, φ is the scoring function for triplets, and $\boldsymbol{I}[\mathrm{X}]$ is the indicator function (1 if condition X is true, and 0 otherwise).

A.1 Mean Reciprocal Rank (MRR)

The Mean Reciprocal Rank(MRR) is defined as:

$$MRR = \frac{1}{2\left|\mathcal{T}\right|}\sum_{i=1}^{\left|\mathcal{T}\right|}\left(\frac{1}{rank_i^{left}} + \frac{1}{rank_i^{right}}\right)$$

A.2 Hits@N

The Hits@n is defined as:

$$Hits@n = \frac{1}{2\left|\mathcal{T}\right|}\sum_{i=1}^{\left|\mathcal{T}\right|}\left(\boldsymbol{I}\left(rank_i^{left} \leq n\right) + \boldsymbol{I}\left(rank_i^{right} \leq n\right)\right)$$