Link Prediction in Knowledge Graphs using Latent Feature Modelling and Neural Tensor Factorisation

Luyolo Magangane and Willie Brink

Stellenbosch University, Stellenbosch, South Africa
luyolo.nqobile@gmail.com, wbrink@sun.ac.za

Abstract. Knowledge graphs can be used to represent interconnected facts about multiple domains as entities (nodes) and relations (edges). The resource description framework (RDF) formalism can be used to encode such facts as subject-predicate-object triples. Link prediction then powers knowledge discovery by scoring possible relationships between entities. Tensor decomposition is an attractive approach to link prediction, as relational domains are usually high-dimensional and sparse; a setting where factorisation methods, particularly the HypER model, have shown very good results. Modern approaches typically also contain nonlinear neural networks to enable the learning of powerful latent representations of entities and relations in a continuous vector space. We introduce optimisations to the training algorithm of HypER, by using batch normalisation to compensate for covariate shift caused by hypernetworks, and propose HypER+. We see similar performance to the HypER baseline on the WN18 dataset, and significant improvement on the FB15k dataset. We then extend our model by initialising entity and relation embeddings with pre-trained word vectors from the GloVe language model, and see further improvements over the baselines on the more challenging WN18RR and FB15k-237 datasets. Our results establish HypER+ as a state-of-the-art model in latent feature modelling based link prediction.

Keywords: Link prediction · Tensor decomposition · Hypernetworks.

1 Introduction

Reasoning over knowledge expressed in natural language is a problem at the forefront of artificial intelligence research. Question answering is a core task of this problem, and is concerned with giving machines the capability of generating an answer given a question.

Knowledge graphs (KGs) model facts as entities (nodes) and relations between entities (edges) [1]. The resource description framework formalism encodes facts as triples of the form "subject-predicate-object", where the subject and object are entities, and the predicate is a relation [2]. An example could be the fact that the subject "Chadwick Boseman" is related to the object "Black Panther"

by the predicate "starred in". Question answering then relies on knowledge discovery; a step-by-step deductive process of inferring new facts from a given set of known facts. Statistical relational learning (SRL) solves the knowledge discovery problem by constructing models with measures of uncertainty in plausible facts not contained in KGs [3].

There has been encouraging progress in SRL to model knowledge in open-domain settings [4]. Link prediction, i.e. inferring plausible relations between KG entities, is now often used as a paradigm for knowledge discovery [5–7]. Latent feature modelling using tensor factorisation [8] is an approach to link prediction that has seen some promising results [9–11]. The first major milestone was demonstrating the bilinear tensor product as a promising model to link prediction [12]. In this model a vector representation of the subject is multiplied with a predicate matrix to produce an entity-relational vector, and the inner product with an object vector then provides a relational plausibility score for the subject and object entities. The second milestone was an integration of the bilinear tensor product and neural networks [13], effectively extending linear tensor factorisation techniques to nonlinear techniques and also enabling the use of pre-trained word embeddings to initialise entity and relation vectors. Third was the use of complex valued embeddings for link prediction [14], to effectively capture antisymmetric relations. Up until that point research had focused on minimising the parameterisation of models. Convolutional networks are parameter efficient, and progress was again realised with the use of deep convolutional models for link prediction [15]. The HypER model [16] introduced the use of hypernetworks (a class of meta-networks trained to configure a main network [17]) to transform relational vectors into relation-specific convolutional filters used by a main network.

In this work we first introduce training algorithm optimisations to a TensorFlow reimplementation of a baseline neural tensor network (NTN) model [18]. We then adapt the HypER model by applying batch normalisation, in an effort to mitigate covariate shift in its hypernetwork, and call the new model HypER+. Finally, in an attempt to leverage semantic information from very large text corpora, we integrate pre-trained word vectors into the HypER+ training process to initialise entity and relational vectors in place of standard random initialisation. Our experimental results on benchmark datasets establish HypER+ as a state-of-the-art model in latent feature modelling based link prediction.

2 Related Work

Statistical relational learning is comprised of three paradigms: latent feature modelling, graph modelling and inductive probabilistic logic programming. In this paper we focus on latent feature modelling approaches, and refer the reader to [1] for a review of the other two. Techniques based on tensor factorisation have become popular in latent feature modelling based link prediction, and decompose relational data represented as a tensor of relational scores to generate a set of constituent vectors.

The RESCAL model by Nickel et al. [12] decomposes a relational score tensor S into an entity matrix $E \in \mathbb{R}^{n \times r}$, a relational tensor $R \in \mathbb{R}^{r \times r \times m}$ and the transpose of the entity matrix $E^T \in \mathbb{R}^{r \times n}$, as follows:

$$S \approx ERE^T. \tag{1}$$

The entity matrix E is composed of entity vectors $e_i \in \mathbb{R}^{1 \times r}$, and the relational tensor R is composed of full-rank matrix slices $W_k \in \mathbb{R}^{r \times r}$, where $k \in \{1, \ldots, m\}$ is the set of KG relations. The vector-matrix product between entity e_i and relational matrix W_k generates an entity-relational vector $h_{i,k} \in \mathbb{R}^{1 \times r}$. The inner product between $h_{i,k}$ and entity vector $e_j \in \mathbb{R}^{r \times 1}$ generates a relational score $s_{i,k,j}$ indicating the plausibility of entities i and j being linked by relation k. In this framework the entities and relations have latent vector representations computed using a parameterised function, and the idea would be for the model to learn optimal parameters from data. Training RESCAL amounts to minimising a squared error between known facts and model predictions.

The TransE model by Bordes et al. [19] follows a premise that many KG facts are presented in hierarchies, and that a translation of the subject vector by the relation should produce an embedding close to the object. The DistMul model by Yang et al. [20] first transforms the subject and object vectors into low-dimensional representations, and then applies a bilinear tensor product using a diagonal relation matrix. This approach effectively models a subset of the entity-relational interactions of RESCAL, and relies on its dimensionality reduction to generate sufficient semantic information. The ComplEx model by Trouillon et al. [14] represents entities with complex vectors, and relations with complex diagonal matrices, to allow the modelling of antisymmetric interactions (such as the predicate "starred in", for example) where the subject and object are not interchangeable. The mentioned approaches all rely on linear tensor factorisation for latent feature modelling. They scale well with large datasets but are limited in their expressiveness.

The neural tensor network (NTN) model by Socher et al. [13] extends the bilinear tensor product in two ways. Firstly, it makes use of a recursive network to score a composition between two entities, and adds that score to the output of the bilinear tensor product to produce a relational score. Secondly, the computed relational score is passed through a squashing nonlinearity in order to generate a measure of confidence in a potential relationship between the two entities. These extensions enable higher levels of expressiveness and increase prediction performance significantly.

Hohenecker and Lukasiewicz [21] extended the NTN model by pre-computing an object representation as an aggregation of all facts in which the object plays a part, and called the new model a relational NTN. The HolE model by Nickel et al. [11] uses a circular correlation during relation prediction. This operation is performed between a subject vector e_i and an object vector e_j, by sliding the object over the subject, and an inner product with the predicate vector then produces a relational score. The ConvE model by Dettmers et al. [15] introduces the convolutional operator in entity-relational modelling. A subject vector $e_i \in \mathbb{R}^r$

and a predicate vector $w_k \in \mathbb{R}^r$ are reshaped individually to $\overline{e_i} \in \mathbb{R}^{r_w \times r_h}$ and $\overline{w_k} \in \mathbb{R}^{r_w \times r_h}$, and then vertically concatenated to create an entity-relational matrix $M \in \mathbb{R}^{r_w \times 2r_h}$. Convolutions are performed on M using a set of trainable filters, shared among all subject-predicate combinations. This creates a feature map that is flattened and passed through a fully-connected layer with a nonlinearity. Inner products are taken with all object vectors to create relational scores, which can be converted to probabilities with the sigmoid function. The convolutional operator increases expressiveness by modelling entity-relation feature interactions across the entire line of concatenation.

The HypER model by Balažević et al. [16] extends convolutional tensor factorisation by using a hypernetwork [17] to generate 2D relation-specific convolutional filters as predicate matrices. A 2D convolution is then taken between a subject vector and predicate matrix, the resulting feature map is reshaped and passed through a fully-connected layer with a nonlinearity, and an inner product leads to relational scores. The hypernetwork may introduce covariate shift [22], or a change in the distribution of inputs to a current layer, due to the simultaneous update of current layer weights and previous layer weights during training. Covariate shift slows down training and may degrade model performance during inference.

We aim to further improve link prediction performance of HypER on standard benchmark datasets, by optimising various aspects of training. To this end, we update the base NTN training algorithm with early stopping, Adam optimisation [23] and hyperparameter random search [24]. We then address the potential covariate shift introduced by hypernetworks, and also use pre-trained word vectors for initialising entity and relational vector embeddings.

3 Neural Tensor Factorisation

Tensor factorisation has shown promise in domains that are high-dimensional and sparse, and lends itself to efficient GPU computation. In the context of KGs, relational score tensors are decomposed into an entity matrix, a relational tensor and a transposed entity matrix. The entity matrix is composed of latent entity representations generated with trainable parameters. The relational tensor is composed of relational matrix slices, also generated with trainable parameters. In practice the bilinear tensor product between the entity matrix, the relational tensor and the transposed entity matrix leads to unnormalised relational scores for potential triples, from which a link can be predicted as the argmax over those scores.

Early tensor factorisation approaches focused on limiting the total number of model parameters in order to scale to large KGs, often at the expense of sufficiently complex entity-relational interaction modelling. In this section we expound on recent significant milestones in latent feature modelling approaches to link prediction that take advantage of the expressiveness of neural compositional models. We also discuss our proposed improvements.

3.1 Neural Tensor Networks

Socher et al. [13] introduced the use of recursive entity representations in the composition of the relational score. Recursive networks try to capture the rules for word combinations by constructing a composition tree between words, in a sequence of words. The NTN model takes advantage of these compositional rules by adding them to the bilinear tensor product as follows:

$$g_{i,k,j} = u_k^T \, f\Big(e_i^T W_k^{[1:m]} e_j + V_k \begin{bmatrix} e_i \\ e_j \end{bmatrix} + b_k\Big). \tag{2}$$

Here $g_{i,k,j}$ is the relational score tensor for subject i, object j and relation k, $u_k \in \mathbb{R}^{m\times 1}$ is an output layer of trainable weights, and f is the tanh activation function. $e_i^T W_k^{[1:m]} e_j$ is the bilinear tensor product between subject vector $e_i \in \mathbb{R}^{r\times 1}$ and object vector $e_j \in \mathbb{R}^{r\times 1}$, for relations $W_k \in \mathbb{R}^{r\times r}$, and b_k is a trainable bias vector. $V_k[\, e_i^T \quad e_j^T \,]^T$ is the recursive entity composition, with $V_k \in \mathbb{R}^{m\times 2r}$ a matrix of trainable parameters.

During training a contrastive max-margin loss is minimised. This loss incorporates the score $g(T^{(n)})$ of a correct sample (a fact present in the KG) and the score $g(T_c^{(n)})$ of a corrupt sample (a randomly generated fact not present in the KG), as follows:

$$J(\Omega) = \sum_{n=1}^{N} \sum_{c=1}^{C} \max\left(0, 1 - g(T^{(n)}) + g(T_c^{(n)})\right) + \lambda ||\Omega||_2^2, \tag{3}$$

where Ω contains all the trainable parameters, N is the number of training samples, and C is the number of randomly corrupted facts to be used per true fact. The hyperparameter λ controls the importance of the ridge regulariser in this loss.

Doss et al. [18] reimplemented the NTN model in TensorFlow. This reimplementation underperforms compared to the original model (as we also find in Section 4.3), relying on AdaGrad optimisation and the same hyperparameters. In an attempt to improve the performance of the reimplementation we apply early stopping, the more modern Adam optimisation algorithm [23] as well as hyperparameter random search [24]. Early stopping tries to prevent overfitting, by tracking performance on the validation set during training and halting the training process as soon as it decreases significantly. Adam is a gradient based stochastic optimisation algorithm that tries to compensate for the sparse gradient signal problem, by incorporating first- and second-order moments of the gradient and parameter-specific adaptive learning rates. Hyperparameter random search defines intervals for all model hyperparameters and randomly samples from those intervals for a set number of training runs, thus taking advantage of the hypothesis that for many datasets only a subset of hyperparameters contribute meaningful variance in model performance, and eliminating the need for a costly exhaustive search.

3.2 Convolutional Networks

The ConvE model of Dettmers et al. [15] introduces the convolutional operator to neural tensor factorisation Specifically, this operator increases expressiveness in entity-relation interaction modelling by using 2D convolutions, which have been found to be particularly effective at modelling the interactions of entities involved in a large number of relations. This may be due to filter parameter sharing between entity-relational combination features, but perhaps more pertinently, the convolutional operator captures a larger variety of entity-relational feature interactions when summarising regions between the respective representations, whereas the bilinear tensor product performs interaction modelling using a simple inner product.

ConvE concatenates subject and predicate matrices along the row axis, and takes convolutions with trainable filters to produce feature maps. The feature maps are flattened and passed through a fully-connected layer with ReLU activation, to generate a latent vector representation. The inner product of this vector and the object vector then leads to an unnormalised relational score for the triple, which is passed through softmax normalisation.

During training a cross-entropy loss is minimised. This loss function is particularly appropriate as we expect only a single object to belong to the fact (subject-predicate-object) and generate a probability close to 1, and every other object to not belong to the fact, generating a probability close to 0.

3.3 Hypernetworks

The HypER model of Balažević et al. [16] extends convolutional tensor factorisation by using a hypernetwork [17] to generate 2D relation-specific convolutional filters as predicate matrices. A hypernetwork is a meta-network that generates parameters for a main network. It compresses those parameters into an input embedding vector, which is analogous to encoding a main network configuration and effectively reduces the total number of parameters without sacrificing performance. The embedding parameters of the hypernetwork are learned during end-to-end training of the main network.

HyPER makes use of a hypernetwork to generate relation-specific convolutional filters, where the subject and predicate filter are used in a convolution operation to generate a subject-predicate feature map. The feature map is flattened into a vector and passed through a fully-connected layer which outputs a hidden vector that is passed through a ReLU nonlinearity. An inner product is then taken with the object vector to produce an unnormalised relational score, and finally a softmax normalisation is applied to generate probabilities for potential relationships between pairs of entities.

We make the observation that hypernetworks may suffer from covariate shift. The hypernetwork generates relational filters for the main network from relational embedding inputs. During training, the distribution of the latent parameters of the relational embeddings could change, altering the inputs used to

generate the filters. This change in distribution may make it hard for the hypernetwork's fully-connected layer to learn the most useful parameters for creating relational filters. Moreover, since the relational filters are used early (upstream) in the main network, the effect of their suboptimal weights might be amplified. To address this issue, we introduce relational input batch normalisation [22]. This operation performs normalisation on relational input batches during training, such that each batch has zero mean and unit variance, to regulate hypernetwork input and prevent covariate shift.

We refer to our improved model, which incorporates the training optimisations mentioned at the end of Section 3.1 as well as batch normalisation in the hypernetwork, as HypER+.

3.4 Pre-trained Word Vectors

As an additional potential improvement we also incorporate pre-trained word vectors from the GloVe language model [25]. Language models capture statistical correlations between words in a language, and in so doing attempt to build an understanding of the semantics of that language. Word meaning is embedded (or encoded) in a vector space, where words that share semantic meaning may be collocated. The GloVe model generates word vectors from both local and global co-occurrence statistics of words in a text corpus, and has been found to outperform other popular models (like Word2Vec).

KGs are comprised of vocabularies of entities and relations. These vocabularies contain a KG word to ID map, which is used during model training to identify the respective entity or relation. A pre-trained GloVe language model is a map of words to vectors, and can be used to look up the corresponding vectors for entities or relations in the KG.

Instead of the standard random initialisation of word vector embeddings, we experiment with using pre-trained GloVe word vectors to initialise 200-dimensional entity and relation embeddings. The respective embeddings are generated by aggregating the set of vectors corresponding to the sequence of words that describe them (where a pre-trained vector does not exist for a particular word, a randomly initialised vector can be generated in its place). This process generates two KG pre-trained vector to ID maps, one for entities and one for relations, and these maps are used to initialise the model entity and relation embeddings respectively. The embeddings are trainable, and updated during end-to-end training of HypER+ to generate representations specific to the KG under consideration.

4 Experiments

We proceed with an empirical examination of the three contributions of this work, namely (1) the application of training optimisation to neural tensor networks, (2) compensating for covariate shift in hypernetworks used for convolutional tensor factorisation, and (3) the initialisation of entity and relation embeddings with pre-trained word vectors.

We evaluate our models on three pairs of progressively more challenging benchmark link prediction datasets (detailed in Section 4.1), and compare a number of performance metrics (explained in Section 4.2) against existing and current state-of-the-art models (Section 4.3). The different datasets illuminate specific aspects, and informal experimentation with exhaustive training of all models on all datasets suggests trends similar to those reported in this paper.

All code needed to train and test our models, and reproduce the results in this section, are available at https://github.com/xhosaBoy.

4.1 Benchmark Datasets

Two of the earliest datasets used to evaluate link prediction are called WN11 and FB13, and were extracted from WordNet [26] and Freebase [27], respectively. WordNet is a lexical database for English, containing a taxonomy with hypernym relationships ("is a") and synonym sets. Freebase is a large collaborative knowledge base composed mainly by community members. WN11 contains just over 125,000 triples, split into training, validation and test sets as indicated in Table 1, with 38,696 unique entities (subjects and objects, each corresponding to a WordNet synset) and 11 unique relations (predicates). FB13 has close to 350,000 triples in total, with 75,043 unique entities and 13 relations.

The WN18 and FB15k datasets were introduced later by Bordes et al. [19]. WN18 consists of about 150,000 triples, split into training, validation and tests sets as indicated in Table 1, and includes 18 unique relations for about 41,000 unique entities from WordNet. FB15k has almost 600,000 triplets, consisting of 14,951 unique entities and 1,345 unique relations.

It was found that WN18 and FB15k both suffer from significant label leakage through inverse relations [15, 28], that is to say many triples in the test sets occur in inverted form in the training sets, making it easy for simple models to do well. Datasets WN18RR [15] and FB15k-237 [28] were created in an effort to avoid such leakage, through the removal of inverse relations from WN18 and FB15k, respectively. WN18RR has just over 93,000 triples over about 41,000 entities and 11 relations, while FB15k-237 has about 310,000 triples over 14,541 entities and 237 relations.

Table 1. The number of triples in the training, validation and tests sets of the three pairs of benchmark datasets considered in this study, along with the number of unique entities and unique relations in each.

Dataset	Train	Val	Test	Entities	Relations
WN11 [26]	112,581	2,609	10,544	38,696	11
FB13 [27]	316,232	5,908	23,733	75,043	13
WN18 [19]	141,442	5,000	5,000	40,943	18
FB15k [19]	483,142	50,000	59,071	14,951	1,345
WN18RR [15]	86,835	3,034	3,134	40,943	11
FB15k-237 [28]	272,115	17,535	20,466	14,541	237

4.2 Performance Metrics

We report a set of standard performance metrics used in the link prediction literature, namely Hit@10, Hit@3, Hit@1 and Mean Reciprocal Rank (MRR). Hit@k (or H@k for short) measures the fraction of times the correct label occurs in the top k predictions, if outputs are ordered by confidence scores. MRR is the average predicted inverse rank of correct labels. For example, if the true label is predicted as the third most likely output, the predicted inverse rank for that sample is 1/3.

All of these metrics will be expressed as percentages, and measured over the respective test sets of the various benchmark datasets, using only the final versions of models (after hyperparameter selection and training).

4.3 Results

For the first set of experiments we use the WN11 and FB13 datasets for training, validation and testing. We compare the test prediction accuracy (Hit@1) achieved by the original NTN model, as reported by Socher et al. [13], with the TensorFlow reimplementation [18] and our optimised version that incorporates early stopping, the Adam optimiser and hyperparamter random search.

Results are presented in Table 2. Our model outperforms the TensorFlow reimplementation, and quite significantly on the WN11 dataset. Training loss curves suggest that hyperparameter random search is mostly responsible for this improvement, as our model begins to outperform the baseline from the first training epoch. Reported results of the original NTN model are still far superior, possibly due to other (unknown) training algorithm modifications and optimisations.

Table 2. Link prediction accuracies (as percentages) achieved by the indicated models on the WN11 and FB13 test sets.

Model	**WN11**	**FB13**
Original NTN model [13]	86.2	90.0
TensorFlow reimplemented NTN [18]	56.2	53.5
Optimised (ours)	67.4	54.8

For the second set of experiments we use the WN18 and FB15k datasets for training, validation and testing. In Table 3 we compare the test set performance of our HypER+ model against HypER and also a suite of previous state-of-the-art models. Two models not mentioned in Section 2 are included here for completeness, namely TorusE [6] and R-GCN [29]. TorusE is an extension of the TransE model that circumvents a certain regularisation problem, while R-GCN models relational data with graph convolutional networks.

On the WN18 dataset our HypER+ model achieves results very close to the HypER model, across all metrics. The R-GCN model achieves the highest

Table 3. Link prediction results on the WN18 and FB15k test sets, as achieved by the indicated models. Our HypER+ model is the bottom row, and best results per column are shown in bold.

Model	**WN18**				**FB15k**			
	H@10	H@3	H@1	MRR	H@10	H@3	H@1	MRR
TransE [19]	89.2	-	-	-	47.1	-	-	-
DistMul [20]	93.6	91.4	72.8	82.2	82.4	73.3	54.6	65.4
ComplEx [14]	94.7	93.6	93.6	94.1	84.0	75.9	59.9	69.2
R-GCN [29]	**96.4**	92.9	69.7	81.9	84.2	76.0	60.1	69.6
TorusE [6]	95.4	95.0	94.3	94.7	83.2	77.1	67.4	73.3
ConvE [15]	95.6	94.6	93.5	94.3	83.1	72.3	55.8	65.7
HypER [16]	95.8	**95.5**	**94.7**	**95.1**	88.5	82.9	73.4	79.0
HypER+ (ours)	95.7	95.4	94.6	95.0	**89.4**	**85.6**	**79.0**	**82.9**

Hit@10 performance, but performs relatively poorly across the other metrics (especially Hit@1). On the FB15k dataset our HypER+ model outperforms all other models significantly. The impact of batch normalisation in the hypernetwork is pronounced, perhaps due to the upstream influence of predicate input covariate shift.

For the third and final set of experiments we use the WN18RR and FB15k-237 datasets for training, validation and testing. These datasets are more challenging subsets of the previous two (WN18 and FB15k), with inverse relations removed to prevent test leakage. Here we compare two versions of our HypER+ model with previous state-of-the-art models: one with randomly initialised embedding vectors and one initialised with pre-trained GloVe word vectors. Results are listed in Table 4.

Our HypER+ model with pre-trained GloVe initialisation leads to state-of-the-art performance across both WN18RR and FB15k-237. The pre-trained embeddings have a particularly pronounced impact on the Hit@10 metric over WN18RR, perhaps due to the smaller number samples in this dataset, but a diminished impact on the Hit@1 metric. Overall it seems as if the pre-trained

Table 4. Link prediction results on the more challenging WN18RR and FB15k-237 test sets, as achieved by the indicated models. Our original HypER+ model and one that was initialised at training time with pre-trained GloVe word vectors form the last two rows, and best results per column are shown in bold.

Model	**WN18RR**				**FB15k-237**			
	H@10	H@3	H@1	MRR	H@10	H@3	H@1	MRR
DistMul [20]	49.0	44.0	39.0	43.0	41.9	26.3	15.5	24.1
ComplEx [14]	51.0	46.0	41.0	44.0	42.8	27.5	15.8	24.7
ConvE [15]	52.0	44.0	40.0	43.0	50.1	35.6	23.7	32.5
HypER [16]	52.2	47.7	43.6	46.5	52.0	37.6	25.2	34.1
HypER+ (ours)	51.9	47.9	**43.8**	46.6	51.6	36.8	24.5	33.5
GloVe init (ours)	**57.8**	**49.3**	43.5	**48.0**	**52.5**	**37.9**	**25.5**	**34.5**

embeddings may provide a more meaningful contribution than batch normalisation in the hypernetwork, as suggested by the inferior performance of HypER+ compared to HypER. Somewhat strangely, this is inconsistent with training and validation accuracies, where HypER+ consistently outperformed HypER. The fact that we use the quoted HypER test results, as opposed to reimplemented and verified test results, may be a partial explanation. That inconsistency aside, pre-trained embeddings from the GloVe language model appear to be effective at improving link prediction performance.

Figure 1 shows the Hit@1 test set accuracy for each of 10 relations from WN18RR, as achieved by our HypER+ model initialised with pre-trained word vectors. The model performs well with synonym relation types, but poorly with compositional and hierarchical relations. This may be due to the inherent similarity and analogy in concepts, whereas compositions and hierarchies can be defined by strict rules. Perhaps, more simply, it could be due to the number of test set samples for each relation, where higher numbers increase the prediction error rate.

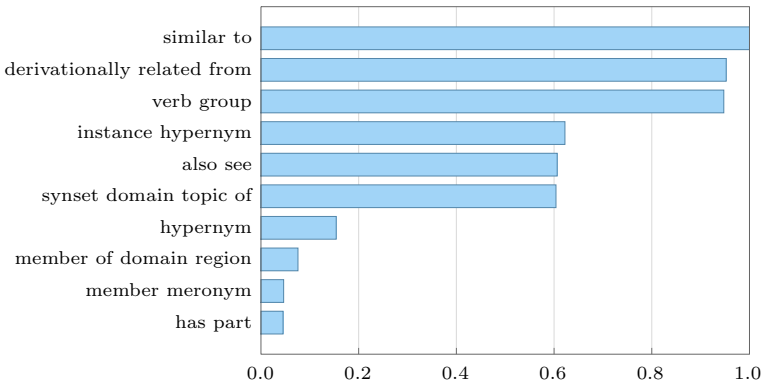

Fig. 1. Hit@1 prediction accuracy per predicate, as achieved by the HypER+ model with pre-trained GloVe word vectors on the WN18RR test set. It should be noted that the predicates are not evenly distributed in the dataset; "hypernym" and "derivationally related from" respectively account for about 40% and 35% of all triples in WN18RR.

For illustrative purposes we show in Table 5 a handful of incorrect test set predictions from our HypER+ model with pre-trained word vectors. These examples are all for the predicate "has part", which according to Figure 1 seems difficult to model accurately. Our model does demonstrate basic conceptual understanding, as most of these mistakes could be deemed forgivable. We would therefore expect reasonable knowledge discovery utility from the model, when used jointly with information retrieval for open-domain question answering.

Table 5. Examples of incorrect predictions made by our HypER+ model with pre-trained word vectors on the WN18RR test set.

Subject	**Predicate**	**Target object**	**Predicted object**
usa	has part	colorado	missouri river
spain	has part	cadiz	jerez de la frontera
electromagnetic spectrum	has part	actinic ray	radio spectrum
systema respiratorium	has part	respiratory tract	respiratory organ
africa	has part	nigeria	senegal
antigen	has part	substance	epitope
amphitheatre	has part	theatre	tiered seat
indian ocean	has part	mauritius	antarctic ocean

5 Conclusions

Neural tensor factorisation has shown promise in extending the performance of latent feature modelling based link prediction in knowledge graphs. The original insight to apply tensor decompositions to relational modelling gave rise to a number of increasingly performant models, and ultimately to our proposed HypER+ model.

There remains an open question as to the extent to which representation structure can continue to improve performance. HypER [16] for example extends the concept of multi-dimensional inputs proposed by ConvE [15], by generating relational matrices as opposed to relying on relational vectors. A natural further extension is generating entity matrices, and then potentially tensor representations for both. Such dense representations may better capture entity and relational concepts. The application of alternative entity-relation feature interaction operators is another natural extension. It has been demonstrated [14, 11, 15] that the Hermitian dot product, circular correlation and convolution operators have potential. An as yet unexplored avenue is the use of stochastic operators that directly encode uncertainty, and may produce relational score probabilities with more appropriate confidence measures.

Some progress in link prediction was recently achieved using the graph modelling paradigm. State-of-the-art Hit@1 accuracy of 46% was achieved on FB15k-237 by Nathani et al. [30], compared to an accuracy of 25.5% achieved by our HypER+ model with pre-trained word vectors. Such a staggering improvement is however not realised on WN18RR, with the model of Nathani et al. achieving a Hit@1 accuracy of 36.1% compared to our model's 43.5%. But these graph modelling approaches clearly demonstrate potential in pushing link prediction performance forward. Their strength could be exploiting a similar idea to the one used to construct the GloVe language model, incorporating both local and global context. Inductive probabilistic logic programming (IPLP) also shows promise [31, 32], although research in this paradigm is much more sparse. It would seem graph modelling, as opposed to latent feature modelling, may provide the biggest contribution to link prediction in the near-term.

References

1. Nickel, M., Murphy, K., Tresp, V., Gabrilovich, E.: A review of relational machine learning for knowledge graphs. Proceedings of the IEEE **104**(1), 11–33 (2016)
2. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia - a crystallization point for the web of data. Journal of Web Semantics **7**(3), 154–165 (2009)
3. Getoor, L., Taskar, B. (eds.): Introduction to Statistical Relational Learning. MIT Press (2007)
4. Chen, D., Fisch, A., Weston, J., Bordes, A: Reading Wikipedia to answer open-domain questions. In: Meeting of the Association for Computational Linguistics, pp. 1870–1879 (2017)
5. Kristiadi, A., Khan, M., Lukovnikov, D., Lehmann, J., Fischer, A.: Incorporating literals into knowledge graph embeddings. In: International Semantic Web Conference, pp. 347–363 (2019)
6. Ebisu, T., Ichise, R.: TorusE: knowledge graph embedding on a Lie group. In: AAAI Conference on Artificial Intelligence, pp. 1819–1826 (2018)
7. Nguyen, D., Nguyen, T., Nguyen, D., Phung, D.: A novel embedding model for knowledge base completion based on convolutional neural network. In: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 327–333 (2018)
8. Kolda, T., Bader, B.: Tensor decompositions and applications. SIAM Review **51**(3), 455–500 (2009)
9. Bordes, A., Weston, J., Collobert, R., Bengio, Y.: Learning structured embeddings of knowledge bases. In: AAAI Conference on Artificial Intelligence, pp. 301–306 (2011)
10. Jenatton, R., Roux, N., Bordes, A., Obozinski, G.: A latent factor model for highly multi-relational data. In: Advances in Neural Information Processing Systems, pp. 3167–3175 (2012)
11. Nickel, M., Rosasco, L., Poggio, T.: Holographic embeddings of knowledge graphs. In: AAAI Conference on Artificial Intelligence, pp. 1955–1961 (2016)
12. Nickel, M., Tresp, V., Kriegel, H.: A three-way model for collective learning on multi-relational data. In: International Conference on Machine Learning, pp. 809–816 (2011)
13. Socher, R., Chen, D., Manning, C., Ng, A.: Reasoning with neural tensor networks for knowledge base completion. In: Advances in Neural Information Processing Systems, pp. 926–934 (2013)
14. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: International Conference on Machine Learning, pp. 2071–2080 (2016)
15. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2D knowledge graph embeddings. In: AAAI Conference on Artificial Intelligence, pp. 1811–1818 (2018)
16. Balažević, I., Allen, C., Hospedales, T.: Hypernetwork knowledge graph embeddings. In: International Conference on Artificial Neural Networks, pp. 553–565 (2019)
17. Ha, D., Dai, A., Le, Q.: Hypernetworks. In: International Conference on Learning Representations (2017)
18. Doss, D., LeNail, A., Liu, C. (2015). Reimplementing neural tensor networks for knowledge base completion in the TensorFlow framework. https://github.com/

dddoss/tensorflow-socher-ntn/blob/master/notes/paper.pdf. Last accessed 15 Sep 2020
19. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems, pp. 2787–2795 (2013)
20. Yang, B., Yih, S., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: International Conference on Learning Representations (2015)
21. Hohenecker, P., Lukasiewicz, T.: Ontology reasoning with deep neural networks. Journal of Artificial Intelligence Research **68**, 503–540 (2020)
22. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate Shift. In: International Conference on Machine Learning, pp. 448–456 (2015)
23. Kingma, D., Ba, J.: Adam: a method for stochastic optimization. In: International Conference on Learning Representations (2015)
24. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research **13**(Feb), 281–305 (2012)
25. Pennington, J., Socher, R., Manning, C.: GloVe: global vectors for word representation. In: Conference on Empirical Methods in Natural Language Processing, pp. 1532–1543 (2014)
26. Miller, G.: WordNet: a lexical database for English. Communications of the ACM **38**(11), 39–41 (1995)
27. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: ACM International Conference on Management of Data, pp. 1247–1250 (2008)
28. Toutanova, K., Chen, D.: Observed versus latent features for knowledge base and text inference. In: Workshop on Continuous Vector Space Models and their Compositionality, pp. 57–66 (2015)
29. Schlichtkrull, M., Kipf, T., Bloem, P., Van Den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: European Semantic Web Conference, pp. 593–607 (2018)
30. Nathani, D., Chauhan, J., Sharma, C., Kaul, M.: Learning attention-based embeddings for relation prediction in knowledge graphs. In: Meeting of the Association for Computational Linguistics, pp. 4710–4723 (2019)
31. Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., Zhang, W.: Knowledge Vault: a web-scale approach to probabilistic knowledge fusion. In: ACM International Conference on Knowledge Discovery and Data Mining, pp. 601–610 (2014)
32. Manhaeve, R., Dumancic, S., Kimmig, A., Demeester, T., De Raedt, L.: DeepProbLog: neural probabilistic logic programming. In: Advances in Neural Information Processing Systems, pp. 3749–3759 (2018)