Link Prediction Using Multi Part Embeddings

Sameh K. Mohamed[1,2,3(✉)] and Vít Nováček[1,2,3]

[1] Data Science Institute, Galway, Ireland
[2] Insight Centre for Data Analytics, Galway, Ireland
{sameh.kamal,vit.novacek}@insight-centre.org
[3] National University of Ireland Galway, Galway, Ireland

Abstract. Knowledge graph embeddings models are widely used to provide scalable and efficient link prediction for knowledge graphs. They use different techniques to model embeddings interactions, where their tensor factorisation based versions are known to provide state-of-the-art results. In recent works, developments on factorisation based knowledge graph embedding models were mostly limited to enhancing the ComplEx and the DistMult models, as they can efficiently provide predictions within linear time and space complexity. In this work, we aim to extend the works of the ComplEx and the DistMult models by proposing a new factorisation model, TriModel, which uses three part embeddings to model a combination of symmetric and asymmetric interactions between embeddings. We perform an empirical evaluation for the TriModel model compared to other tensor factorisation models on different training configurations (loss functions and regularisation terms), and we show that the TriModel model provides the state-of-the-art results in all configurations. In our experiments, we use standard benchmarking datasets (WN18, WN18RR, FB15k, FB15k-237, YAGO10) along with a new NELL based benchmarking dataset (NELL239) that we have developed.

Keywords: Knowledge graph embedding · Link prediction

1 Introduction

In recent years, knowledge graph embedding (KGE) models have witnessed rapid developments that have allowed them to excel in the task of link prediction for knowledge graphs [22]. They learn embeddings using different techniques like tensor factorisation, latent distance similarity and convolutional filters in order to rank facts in the form of (subject, predicate, object) triples according to their factuality. In this context, their tensor factorisation based versions like the DistMult [23] and the ComplEx [21] models are known to provide state-of-the-art results within linear time and space complexity [22]. The scalable and efficient predictions achieved by these models have encouraged researchers to investigate advancing the DistMult and the ComplEx models by utilising different training objectives and regularisation terms [8,9].

P. Hitzler et al. (Eds.): ESWC 2019, LNCS 11503, pp. 240–254, 2019.
https://doi.org/10.1007/978-3-030-21348-0_16

In this work, our objective is to propose a new factorisation based knowledge graph embedding model that extends the works of the DistMult and the ComplEx models while preserving their linear time and space complexity. We achieve that by modifying two of their main components: the embedding representation, and the embedding interaction function.

While both the DistMult and the ComplEx models use the bilinear product of the subject, the predicate and the object embeddings as an embedding interaction function to encode knowledge facts, they represent their embeddings using different systems. The DistMult model uses real values to represent its embedding vectors, which leads to learning a symmetric representation of all predicates due to the symmetric nature of the product operator on real numbers. On the other hand, the ComplEx model represents embeddings using complex numbers, where each the embeddings of an entity or a relation is represented using two vectors (real and imaginary parts). The ComplEx model also represents entities in the object mode as the complex conjugate of their subject form [21]. This enables the ComplEx model to encode both symmetric and asymmetric predicates.

Since the embeddings of the ComplEx models are represented using two part embeddings (real and imaginary parts), their bilinear product (ComplEx's embedding interaction function) consists of different interaction components unlike the DisMult model with only one bilinear product component. Each of these components is a bilinear product of a combination of real and imaginary vectors of the subject, the predicate and the object embeddings, which gives the ComplEx model its ability to model asymmetric predicates.

In this work, we investigate both the embedding representation and the embedding interaction components of the ComplEx model, where we show that the ComplEx embedding interaction components are sufficient but not necessary to model asymmetric predicates. We also show that our proposed model, TriModel, can efficiently encode both symmetric and asymmetric predicates using simple embedding interaction components that rely on embeddings of three parts. To assess our model compared to the ComplEx model, we carry experiments on both models using different training objectives and regularisation terms, where our results show that our new model, TriModel, provide equivalent or better results than the ComplEx model on all configurations. We also propose a new NELL [12] based benchmarking dataset that contains a small number of training, validation and testing facts that can be used to facilitate fast development of new knowledge graph embedding models.

2 Background and Related Works

Knowledge graph embedding models learn low rank vector representation *i.e.* embeddings for graph entities and relations. In the link prediction task, they learn embeddings in order to rank knowledge graph facts according to their factuality. The process of learning these embeddings consists of different phases.

First, they initialise embeddings using random noise. These embeddings are then used to score a set of true and false facts, where a score of a fact is generated by computing the interaction between the fact's subject, predicate and object embeddings using a model dependent scoring function. Finally, embeddings are updated by a training loss that usually represents a min-max loss, where the objective is to maximise true facts scores and minimise false facts scores.

In this section we discuss scoring functions and training loss functions in state-of-the-art knowledge graph embedding models. We define our notation as follows: for any given knowledge graph, E is the set of all entities, R is the set of all relations *i.e.* predicates, N_e and N_r are the numbers of entities and relations respectively, T is the set of all known true facts, e and w are matrices of sizes $N_e \times K$ and $N_r \times K$ respectively that represent entities and relations embeddings of rank K, ϕ_{spo} is the score of the triple (s, p, o), and $\mathcal{L}$ is the model's training loss.

2.1 Scoring Functions

Knowledge graph embedding models generate scores for facts using model dependent scoring functions that compute interactions between facts' components embeddings. These functions use different approaches to compute embeddings interactions like distance between embeddings [2], embedding factorisation [21] or embeddings convolutional filters [5].

In the following, we present these approaches and specify some examples of knowledge graph embedding models that use them.

• *Distance-based embeddings interactions*: The Translating Embedding model (TransE) [2] is one of the early models that use distance between embeddings to generate triple scores. It interprets triple's embeddings interactions as a linear translation of the subject to the object such that $e_s + w_p = e_o$, and generates a score for a triple as follows:

$$\phi_{spo}^{\text{TransE}} = \|e_s + w_p - e_o\|_{l1/l2}, \tag{1}$$

where true facts have zero score and false facts have higher scores. This approach provides scalable and efficient embeddings learning as it has linear time and space complexity. However, it fails to provide efficient representation for interactions in one-to-many, many-to-many and many-to-one predicates as its design assumes one object per each subject-predicate combination.

• *Factorisation-based embedding interactions*: Interactions based on embedding factorisation provide better representation for predicates with high cardinality. They have been adopted in models like DistMult [23] and ComplEx [21]. The DistMult model uses the bilinear product of embeddings of the subject, the predicate, and the object as their interaction, and its scoring function is defined as follows:

$$\phi_{spo}^{\text{DistMult}} = \sum_{k=1}^{K} e_{s_k} w_{p_k} e_{o_k} \tag{2}$$

where e_{s_k} is the k-th component of subject entity s embedding vector e_s. DistMult achieved a significant improvement in accuracy in the task of link prediction over models like TransE. However, the symmetry of embedding scoring functions affects its predictive power on asymmetric predicates as it cannot capture the direction of the predicate. On the other hand, the ComplEx model uses embedding in a complex form to model data with asymmetry. It models embeddings interactions using the product of complex embeddings, and its scores are defined as follows:

$$\begin{aligned}\phi_{spo}^{\text{ComplEx}} &= \text{Re}(\sum_{k=1}^{K} e_{s_k} w_{p_k} \overline{e}_{o_k}) \\ &= \sum_{k=1}^{K} e_{s_k}^r w_{p_k}^r e_{o_k}^r + e_{s_k}^i w_{p_k}^r e_{o_k}^i \\ &\quad + e_{s_k}^r w_{p_k}^i e_{o_k}^i - e_{s_k}^i w_{p_k}^i e_{o_k}^r\end{aligned} \tag{3}$$

where $Re(x)$ represents the real part of complex number x and all embeddings are in complex form such that $e, w \in \mathbb{C}$, e^r and e^i are respectively the real and imaginary parts of e, and $\overline{e}_o$ is the complex conjugate of the object embeddings e_o such that $\overline{e}_o = e_o^r - ie_o^i$ and this introduces asymmetry to the scoring function. Using this notation, ComplEx can handle data with asymmetric predicates, and to keep scores in the real spaces it only uses the real part of embeddings product outcome. ComplEx preserves both linear time and linear space complexities as in TransE and DistMult, however, it surpasses their accuracies in the task of link prediction due to its ability to model a wider set of predicate types.

• *Convolution-based embeddings interactions*: Following the success of convolutional neural networks image processing tasks, models like R-GCN [17] and ConvE [5] utilized convolutional networks to learn knowledge graph embeddings. The R-GCN model learns entity embeddings using a combination of convolutional filters of its neighbours, where each predicate represent a convolution filter and each neighbour entity represents an input for the corresponding predicate filter. This approach is combined with the DistMult model to perform link prediction. Meanwhile, the ConvE model concatenates subject and predicate embeddings vectors into an image (a matrix form), then it uses a 2D convolutional pipeline to transform this matrix into a vector and computes its interaction with the object entity embeddings to generate a corresponding score as follows:

$$\phi_{spo}^{\text{ConvE}} = f(vec(f([\overline{e_s}; \overline{w_p}] * \omega))W)e_o \tag{4}$$

where $\overline{e_s}$ and $\overline{w_p}$ denotes a 2D reshaping of e_s and w_p, ω is a convolution filter, f denotes a non-linear function, $vec(x)$ is a transformation function that reshape matrix x of size $m \times n$ into a vector of size $mn \times 1$.

2.2 Loss Functions

The task of link prediction can generally be cast as a learning to rank problem where the object is to rank knowledge graph triples according to their factuality. Thus, knowledge graph embedding models traditionally use ranking loss

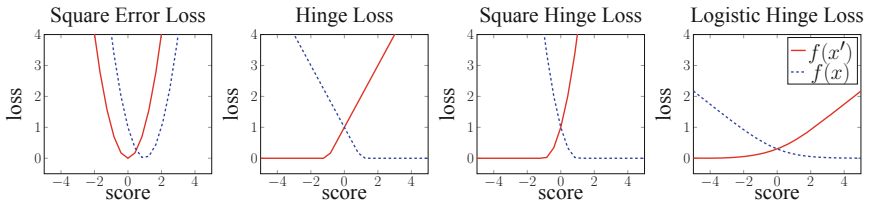

Fig. 1. Plots of different knowledge graph embedding loss functions and their training error slopes. Red lines represent the score/loss slopes of the false triples and blue lines represent the score/loss slopes of the true triples. (Color figure online)

approaches like pairwise and pointwise loss functions as in TransE and ComplEx respectively to model their training loss during the learning process. In these approaches a set of negative facts *i.e.* corruptions, is generated using a uniform random sample of entities to represent false facts, where training loss uses a min-max approach to maximise true facts scores and minimise false facts scores. Meanwhile, recent attempts considered using a multi-class loss to represent training error, where a triple (s, p, o) is divided into an input (s, p) and a corresponding class o and the objective is to assign class o to the (s, p) input.

In the following, we discuss these two approaches with examples from state-of-the-art knowledge graph embedding models.

• *Ranking loss functions*: Knowledge graph embedding models has adopted different pointwise and pairwise ranking losses like hinge loss and logistic loss to model their training loss. Hinge loss can be interpreted as a pointwise loss or a pairwise loss that minimises the scores of negative facts and maximise the scores of positive facts to reach a specific configurable value. This approach is used in HolE [15], and it is defined as:

$$\underset{\text{Hinge}}{\mathcal{L}} = \sum_{x \in X} [\lambda - l(x) \cdot f(x)]_+, \tag{5}$$

where $l(x) = 1$ if x is true and -1 otherwise and $[c]_+$ is equal to $\max(c, 0)$. This effectively generates two different loss slopes for positive and negative scores as shown in Fig. 1.

The squared error loss can also be adopted as a pointwise ranking loss function. For example, the RESCAL [16] model uses the squared error to model its training loss with the objective of minimising the difference between model scores and their actual labels:

$$\underset{\text{SE}}{\mathcal{L}} = \frac{1}{2} \sum_{i=1}^{n} (f(x_i) - l(x_i))^2. \tag{6}$$

The optimal score for true and false facts is 1 and 0, respectively, as shown in Fig. 1. Also, the squared loss requires less training time since it does not require configurable training parameters, shrinking the search space of hyperparameters compared to other losses (*e.g.*, the margin parameter of the hinge loss).

The ComplEx [21] model uses a logistic loss, which is a smoother version of pointwise hinge loss without the margin requirement (*cf.* Fig. 1). Logistic loss uses a logistic function to minimise negative triples score and maximise positive triples score. This is similar to hinge loss, but uses a smoother linear loss slope defined as:

$$\underset{\text{logistic}_{Pt}}{\mathcal{L}} = \sum_{x \in T} \log(1 + \exp(-l(x) \cdot f(x))), \tag{7}$$

where $l(x)$ is the true label of fact x that is equal to 1 for positive facts and is equal to -1 otherwise.

• *Multi-class loss approach*: ConvE model proposed a new binary cross entropy multi-class loss to model its training error. In this setting, the whole vocabulary of entities is used to train each positive fact that for a triple (s, p, o) all facts (s, p, o') with $o' \in E$ and $o' \neq o$ are considered false. Despite the extra computational cost of this approach, it allowed ConvE to generalise over a larger sample of negative assistances therefore surpassing other approaches in accuracy [5]. In a recent work, Lacroix et al. [9] introduced a softmax regression loss to model the training error of the ComplEx model as a multi-class problem. In this approach, the objective for each triple (s, p, o) is to minimise the following losses:

$$\begin{aligned} \mathcal{L}_{spo}^{\log-\text{softmax}} &= \mathcal{L}_{spo}^{o'} + \mathcal{L}_{spo}^{s'}, \\ \mathcal{L}_{spo}^{o'} &= -\phi_{spo} + \log(\textstyle\sum_{o'} \exp(\phi_{spo'}) \\ \mathcal{L}_{spo}^{s'} &= -\phi_{spo} + \log(\textstyle\sum_{s'} \exp(\phi_{s'po}) \end{aligned} \tag{8}$$

where $s' \in E$, $s' \neq s$, $o' \in E$ and $o' \neq o$. This resembles a log-loss of the softmax value of the positive triple compared to all possible object and subject corruptions where the objective is to maximise positive facts scores and minimise all other scores. This approach achieved a significant improvement to the prediction accuracy of ComplEx model over all benchmark datasets [9].

2.3 Ranking Evaluation Metrics

Learning to rank models are evaluated using different ranking measures including Mean Average Precision (MAP), Normalised Discounted Cumulative Gain (NDCG), and Mean Reciprocal Rank (MRR). In this study, we only focus on the Mean Reciprocal Rank (MRR) since it is the main metric used in previous related works.

Mean Reciprocal Rank (MRR). The Reciprocal Rank (RR) is a statistical measure used to evaluate the response of ranking models depending on the rank of the first correct answer. The MRR is the average of the reciprocal ranks of results for different queries in Q. Formally, MRR is defined as:

$$\text{MRR} = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\mathcal{R}(\mathbf{x}_i, f)},$$

Table 1. A comparison between the ComplEx model and different variants of its scoring functions on standard benchmarking datasets

Model	Definition	NELL239		WN18RR		FB237	
		MRR	H@10	MRR	H@10	MRR	H@10
ComplEx	$i_1 + i_2 + i_3 - i_4$	0.35	0.51	0.44	0.51	0.22	0.41
ComplEx-V1	$i_1 + i_2 + i_3$	0.34	0.51	0.45	0.52	0.22	0.40
ComplEx-V2	$i_2 + i_3 + i_4$	0.34	0.50	0.44	0.51	0.21	0.38
ComplEx-V3	$i_1 + i_2 - i_4$	0.34	0.51	0.45	0.52	0.22	0.40
ComplEx-V4	$i_1 + i_3 - i_4$	0.33	0.50	0.45	0.50	0.21	0.39

where x_i is the highest ranked relevant item for query q_i. Values of RR and MRR have a maximum of 1 for queries with true items ranked first, and get closer to 0 when the first true item is ranked in lower positions.

3 The TriModel Model

In this section, we motivate for the design decision of TriModel model, and we present its way to model embeddings interaction and training loss.

3.1 Motivation

Currently, models using factorisation-based knowledge graph embedding approaches like DistMult and ComplEx achieve state-of-the-art results across all benchmarking datasets [9]. In the DistMult model, embeddings interactions are modelled using a symmetric function that computes the product of embeddings of the subject, the predicate and the object. This approach was able to surpass other distance-based embedding techniques like TransE [23]. However, it failed to model facts with asymmetric predicate due to its design. The ComplEx model tackle this problem using a embeddings in the complex space where its embeddings interactions use the complex conjugate of object embeddings to break the symmetry of the interactions. This approach provided significant accuracy improvements over DistMult as it successfully models a wider range of predicates.

The ComplEx embeddings interaction function (defined in Sect. 2) can be redefined as a simple set of interactions of two part embeddings as follows:

$$\phi_{spo}^{\text{ComplEx}} = \sum_k i_1 + i_2 + i_3 - i_4 \tag{9}$$

where $\sum_k$ is the sum of all embeddings components of index $k = \{1, ..., K\}$, and interactions i_1, i_2, i_3 and i_4 are defined as follows:

$$i_1 = e_s^1 w_p^1 e_o^1, \quad i_2 = e_s^2 w_p^1 e_o^2, \quad i_3 = e_s^1 w_p^2 e_o^2, \quad i_4 = e_s^2 w_p^2 e_o^1$$

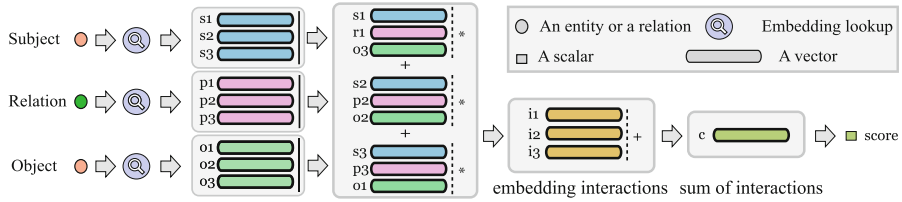

Fig. 2. Visual explanation for the flow of the TriModel model's scoring function, where embedding interactions are represented by $i_1 = e_s^3 w_p^3 e_o^1$, $i_2 = e_s^2 w_p^2 e_o^2$, and $i_3 = e_s^1 w_p^1 e_o^3$.

where e^1 represents embeddings part 1, and e^2 is part 2 ($1 \rightarrow$ real and $2 \rightarrow$ imaginary). Following this notation, we can see that the ComplEx model is a set of two symmetric interaction i_1 and i_2 and two asymmetric interactions i_3 and i_4. Furthermore, this encouraged us to investigate the effect of using other forms of combined symmetric and asymmetric interactions to model embeddings interactions in knowledge graph embeddings. We investigated different combination of interactions i_1, i_2, i_3 and i_4, and we have found that by removing and/or changing the definition of one of these interactions (maintaining that the interactions use all triple components) will preserve similar or insignificantly different prediction accuracy across different benchmarking datasets (See Table 1). This led us to investigate other different forms of interactions that uses a combination of symmetric and asymmetric interactions where we found that using embeddings of three parts can lead to better predictive accuracy than the ComplEx and the DistMult models.

3.2 TriModel Embeddings Interactions

In the TriModel model, we represent each entity and relation using three embedding vectors such that the embedding of entity i is $\{e_i^1, e_i^2, e_i^3\}$ and the embedding of relation j is $\{w_j^1, w_j^2, w_j^3\}$ where e^m denotes the m part of the embeddings and where $m \in 1, 2, 3$ is used to represent the three embeddings parts.

The TriModel model is a tensor factorisation based model, where its embeddings interaction function (scoring function) is defined as follows:

$$\phi_{spo}^{\text{TriPart}} = \sum_{k=1}^{K} e_{sk}^1 w_{pk}^1 e_{ok}^3 + e_{sk}^2 w_{pk}^2 e_{ok}^2 + e_{sk}^3 w_{pk}^3 e_{ok}^1 \tag{10}$$

where k denotes the index of the embedding vector entries. The model uses a set of three interactions: one symmetric interaction: $(e_s^2 w_p^2 e_o^2)$ and two asymmetric interactions: $(e_s^1 w_p^1 e_o^3)$ and $(e_s^3 w_p^3 e_o^1)$ as shown in Fig. 2. This approach models both symmetry and asymmetry in a simple form similar to the DistMult model where the DisMult model can be seen as a special case of the TriModel model if the first and third embeddings part are equivalent ($e^1 = e^3$).

Table 2. Statistics of the benchmarking datasets.

Dataset	Entity count	Relation count	Training	Validation	Testing
WN18	41k	18	141k	5k	5k
WN18RR	41k	11	87k	3k	3k
FB15k	15k	1k	500k	50k	60k
FB15k-237	15k	237	272k	18k	20k
YAGO10	123k	37	1M	5k	5k
NELL239	48k	239	74k	3k	3k

3.3 Training the TriModel Model

Trouillon et al. [20] showed that despite the equivalence of HolE and ComplEx models' scoring functions, they produce different results as they use different loss functions. They concluded that the logistic loss version of ComplEx outperforms its hinge loss version. In addition, we have investigated different other ranking losses with the ComplEx model, and we have found that squared error loss can significantly enhance the performance of ComplEx on multiple benchmarking datasets.

The TriModel model performs its learning process using two different training loss configurations: the traditional ranking loss and the multi-class loss. In the ranking loss configuration, the TriModel model uses the squared error (Eq. 6) and the logistic loss (Eq. 7) to model its training error, where a grid search is performed to choose the optimal loss representation for each dataset. In the multi-class configuration, it uses the negative-log softmax loss (Eq. 8) with the nuclear 3-norm regularisation [9] which is defined as follows:

$$\begin{aligned} \mathcal{L}_{spo}^{\text{TriModel}} = &- \phi_{spo} + \log(\sum\nolimits_{o'} \exp(\phi_{spo'})) \\ &- \phi_{spo} + \log(\sum\nolimits_{s'} \exp(\phi_{s'po})) \\ + \frac{\lambda}{3} &\sum\nolimits_{k=1}^{K} \sum\nolimits_{m=1}^{3} (|e_s^m|^3 + |w_p^m|^3 + |e_o^m|^3) \end{aligned} \tag{11}$$

where m denotes the embedding part index, λ denotes a configurable regularisation weight parameter and $|x|$ is the absolute value of x. This allows the model to answer the link prediction task in both directions: (subject, predicate, ?) and (?, predicate, object). We also consider the use of predicate reciprocals in training as described in Lacroix et al. [9], where inverses of training predicates are

added to the training set and trained with their corresponding original facts as shown in the following:

$$\begin{aligned} \mathcal{L}_{spo}^{\text{TriModel}} &= -\phi_{spo} + \log(\sum\nolimits_{o'} \exp(\phi_{spo'})) \\ &\quad -\phi_{spo} + \log(\sum\nolimits_{s'} \exp(\phi_{o(p+N_r)s})) \\ &\quad + \frac{\lambda}{3} \sum\nolimits_{k=1}^{K} \sum\nolimits_{m=1}^{3} (|e_s^m|^3 + |w_p^m|^3 + |w_{p+N_r}^m|^3 + |e_o^m|^3) \end{aligned} \tag{12}$$

where predicate $p + N_r$ is the inverse of the predicate p where the model learns and evaluates inverse facts using inverses of their original predicates. For all the multi-class configurations, the TriModel model regularises the training facts embeddings using a dropout layer [18] with weighted probability that it learns during the grid search.

4 Experiments

In this section, we discuss the setup of our experiments where we present the evaluation protocol, the benchmarking datasets and our implementation details.

4.1 Data

In our experiments we use six knowledge graph benchmarking datasets:

- WN18 & WN18RR: subsets of the WordNet dataset [11] that contains lexical information of the English language [2,5].
- FB15k & FB15k-237: subsets of the Freebase dataset [1] that contains information about general human knowledge [2,19].
- YAGO10: a subset of the YAGO3 dataset [10] that contains information mostly about people and their citizenship, gender, and professions knowledge [3].
- NELL239: a subset of the NELL dataset [6,12] that we have created to test our model, which contains general knowledge about people, places, sports teams, universities, etc.

Table 2 contains statistics about our experiments' benchmarking datasets[1].

4.2 Implementation

We use TensorFlow framework (GPU) along with Python 3.5 to perform our experiments. All experiments were executed on a Linux machine with processor Intel(R) Core(TM) i70.4790K CPU @ 4.00 GHz, 32 GB RAM, and an nVidia Titan Xp GPU.

[1] All the benchmarking datasets can be downloaded using the following url: https://figshare.com/s/88ea0f4b8b139a13224f.

4.3 Experiments Setup

We perform our experiments in two different configurations:

(1) Ranking loss based learning: the models are trained using a ranking based loss function, where our model chooses between squared error loss and logistic loss using grid search.
(2) Multi-class loss based learning: the models is trained using a multi-class based training functions, where our model uses the softmax negative log loss functions described in Eqs. 11 and 12.

In all of our experiments we initialise our embeddings using the Glorot uniform random generator [7] and we optimise the training loss using the Adagrad optimiser, where the learning rate lr $\in \{0.1, 0.3, 0.5\}$, embeddings size $K \in 50, 75, 100, 150, 200$ and batch size $b \in \{1000, 3000, 5000, 8000\}$ except for YAGO10 where we only use $b \in \{1000, 2000\}$. The rest of the grid search hyper parameters are defined as follows: in the ranking loss approach, we use the negative sampling ratio $n \in \{2, 5, 10, 25, 50\}$ and in the multi-class approach we use regularisation weight $\lambda \in \{0.1, 0.3, 0.35, 0.01, 0.03, 0.035\}$ and dropout $d \in \{0.0, 0.1, 0.2, 0.01, 0.02\}$. The number of training epochs is fixed to 1000, where in the ranking loss configuration we do an early check every 50 epochs to stop training when MRR stop improving on the validation set to prevent over-fitting.

In the evaluation process, we only consider filtered MRR and Hits@10 metrics [2]. In addition, in the ranking loss configuration, TriModel model uses a softmax normalisation of the scores of objects and subjects corruptions, that a score of a corrupted object triple (s, p, o_i) is defined as:

$$\phi_{spo_i} = \frac{\exp(\phi_{spo_i})}{\sum_{o' \in E} \exp(\phi_{spo'})},$$

similarly, we apply a softmax normalisation to the scores of all possible subject entities.

5 Results and Discussion

In this section we discuss findings and results of our experiments shown in Tables 3 and 4, where the experiments are divided into two configurations: models with ranking loss functions and models with multi-class based loss functions.

5.1 Results of the Ranking Loss Configuration

In the results of the ranking loss configuration shown in Table 3, the results show that the TriModel model achieves best results in terms of MRR and hits@10 in five out of six benchmarking datasets with a margin of up to 10% as in the YAGO10 dataset. However, on the FB15k-237 ConvKB [14] retains state-of-the-art results in terms of MRR and Hits@10. Results also show that the factorisation

Table 3. Link prediction results on standard benchmarking datasets. ⋆ Results taken from [21] and our own experiments.

	Model	WN18		WN18RR		FB15k		FB15k-237		YAGO10		NELL239	
		MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
Ranking loss	CP	0.08	0.13	-	-	0.33	0.53	-	-	-	-	-	-
	TransE ⋆	0.52	0.94	0.20	0.47	0.52	0.76	.29	0.48	0.27	0.44	0.27	0.43
	ConvKB	-	-	0.25	0.53	-	-	**0.40**	**0.52**	-	-	-	-
	DistMult ⋆	0.82	0.94	0.43	0.49	0.65	0.82	0.24	0.42	0.34	0.54	0.31	0.48
	ComplEx ⋆	0.94	0.95	0.44	0.51	0.70	0.84	0.22	0.41	0.36	0.55	0.35	0.52
	R-GCN	0.81	0.96	-	-	0.70	0.84	0.25	0.42	-	-	-	-
	TriModel	**0.95**	**0.96**	**0.50**	**0.57**	**0.73**	**0.86**	0.25	0.43	**0.46**	**0.62**	**0.37**	**0.53**

based models like the DistMult, ComplEx, R-GCN and TriModel models generally outperform distance based models like the TransE and ConvKB models. However, on the FB15k-237 dataset, both distance based models outperform all other factorisation based models with a margin of up to 15% in the case of the ConvKB and the TriModel model. We intend to perform further analysis on this dataset compared to other datasets to investigate why tensor factorisation models fail to provide state-of-the-art results in future works.

5.2 Results of the Multi-class Loss Configuration

Results of the multi-class based approach show that TriModel model provide state-of-the-art result on all benchmarking datasets, where the ComplEx models provide equivalent results on 3 out 6 datasets. Our reported results of the ComplEx model with multi-class log-loss introduced by Lacroix et al. [9] are slightly different from their reported results as we re-evaluated their models with restricted embeddings size to a maximum of 200. In their work they used an embedding size of 2000, which is impractical for embedding knowledge graphs in real applications. And other previous works using the TransE, DistMult, ComplEx, ConvE, and ConvKB models have limited their experiments to a maximum embedding size of 200. In our experiments, we limited our embedding size to 200 and we have re-evaluated the models of [9] using the same restriction for a fair comparison[2].

5.3 Ranking and Multi-class Approaches

In the link prediction task, the objective of knowledge graph embedding models is to learn embeddings that rank triples according to their faculty. This is achieved by learning to rank original true triples against other negative triple instances, where the negative instances are modelled in different ways in ranking approaches and multi-class loss approaches.

[2] We have used the code provided at: https://github.com/facebookresearch/kbc for the evaluation of the models: CP-N3, CP-N3-R, ComplEx-N3 and ComplEx-N3-R.

Table 4. Link prediction results on standard benchmarking datasets. † Results taken from [9] with embedding size (K) limited to 200.

	Model	WN18		WN18RR		FB15k		FB15k-237		YAGO10		NELL239	
		MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
Multi-class loss	ConvE	0.94	0.95	0.46	0.48	0.75	0.87	0.32	0.49	0.52	0.66	0.37	0.45
	CP-N3 †	0.12	0.18	0.08	0.14	0.35	0.56	0.22	0.42	0.40	0.64	-	-
	ComplEx-N3 †	0.92	0.95	0.44	0.52	0.58	0.79	0.30	0.51	0.46	0.67	-	-
	CP-N3-R †	0.93	0.94	0.41	0.45	0.62	0.78	0.30	0.47	0.55	0.69	-	-
	ComplEx-N3-R †	**0.95**	**0.96**	**0.47**	**0.54**	0.79	0.88	**0.35**	**0.54**	**0.57**	0.70	-	-
	TriModel - N3	**0.95**	**0.96**	**0.47**	**0.54**	**0.84**	**0.91**	**0.35**	**0.54**	**0.57**	**0.71**	**0.41**	0.57
	TriModel -N3-R	**0.95**	**0.96**	**0.47**	**0.54**	0.81	**0.91**	**0.35**	**0.54**	**0.57**	0.70	**0.41**	**0.58**

In learning to rank approach, models use a ranking loss *e.g.* pointwise or pairwise loss to rank a set of true and negative instances [4], where negative instances are generated by corrupting true training facts with a ratio of negative to positive instances [2]. This corruption happens by changing either the subject or object of the true triple instance. In this configuration, the ratio of negative to positive instances is traditionally learnt using a grid search, where models compromise between the accuracy achieved by increasing the ratio and the runtime required for training.

On the other hand, multi-class based models train to rank positive triples against all their possible corruptions as a multi-class problem where the range of classes is the set of all entities. For example, training on a triple (s, p, o) is achieved by learning the right classes "s" and "o" for the pairs (?, p, o) and (s, p, ?) respectively, where the set of possible class is E of size N_e. Despite the enhancements of the predictions accuracy achieved by such approaches [5,9], they can have scalability issues in real-world large sized knowledge graphs with large numbers of entities due to the fact that they use the full entities' vocabulary as negative instances [13].

In summary, our model provides significantly better results than other SOTA models in the ranking setting, which is scalable and thus better-suited to real-world applications. In addition to that, our model has equivalent or slightly better performance than SOTA models on the multi-class approach.

6 Conclusions and Future Work

In this work, we have presented the TriModel model, a new tensor factorisation based knowledge graph embedding model that represents knowledge entities an relation using three parts embeddings, where its embedding interaction function can model both symmetric and asymmetric predicates. We have shown by experiments that the TriModel model outperforms other tensor factorisation based models like the ComplEx and the DistMult on different training objectives and across all standard benchmarking datasets. We have also introduced a

new challenging small size benchmarking datasets, NELL239, that can be used to facilitate fast development of new knowledge graph embedding models.

In our future works, we intend to investigate new possible approaches to model embedding interactions of tensor factorisation models, and we intend to analyse the effects of properties of knowledge graph datasets like FB15k-237 on the efficiency of tensor factorisation based models.

Acknowledgements. This work has been supported by the TOMOE project funded by Fujitsu Laboratories Ltd., Japan and Insight Centre for Data Analytics at National University of Ireland Galway, Ireland (supported by the Science Foundation Ireland grant 12/RC/2289). The GPU card used in our experiments is granted to us by the Nvidia GPU Grant Program.

References

1. Bollacker, K.D., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: SIGMOD Conference, pp. 1247–1250. ACM (2008)
2. Bordes, A., Usunier, N., García-Durán, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NIPS, pp. 2787–2795 (2013)
3. Bouchard, G., Singh, S., Trouillon, T.: On approximate reasoning capabilities of low-rank vector spaces. In: AAAI Spring Symposium on Knowledge Representation and Reasoning (KRR): Integrating Symbolic and Neural Approaches. AAAI Press (2015)
4. Chen, W., Liu, T., Lan, Y., Ma, Z., Li, H.: Ranking measures and loss functions in learning to rank. In: NIPS, pp. 315–323. Curran Associates, Inc. (2009)
5. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2D knowledge graph embeddings. In: AAAI. AAAI Press (2018)
6. Gardner, M., Mitchell, T.M.: Efficient and expressive knowledge base completion using subgraph feature extraction. In: EMNLP, pp. 1488–1498. The Association for Computational Linguistics (2015)
7. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS. JMLR Proceedings, vol. 9, pp. 249–256. JMLR.org (2010)
8. Kadlec, R., Bajgar, O., Kleindienst, J.: Knowledge base completion: baselines strike back. In: Rep4NLP@ACL, pp. 69–74. Association for Computational Linguistics (2017)
9. Lacroix, T., Usunier, N., Obozinski, G.: Canonical tensor decomposition for knowledge base completion. In: ICML. JMLR Workshop and Conference Proceedings, vol. 80, pp. 2869–2878. JMLR.org (2018)
10. Mahdisoltani, F., Biega, J., Suchanek, F.M.: YAGO3: a knowledge base from multilingual Wikipedias. In: CIDR (2015). www.cidrdb.org
11. Miller, G.A.: WordNet: a lexical database for English. Commun. ACM **38**(11), 39–41 (1995)
12. Mitchell, T.M., et al.: Never-ending learning. Commun. ACM **61**(5), 103–115 (2018)
13. Mnih, A., Kavukcuoglu, K.: Learning word embeddings efficiently with noise-contrastive estimation. In: NIPS, pp. 2265–2273 (2013)

14. Nguyen, D.Q., Nguyen, T.D., Nguyen, D.Q., Phung, D.Q.: A novel embedding model for knowledge base completion based on convolutional neural network. In: NAACL-HLT (2), pp. 327–333. Association for Computational Linguistics (2018)
15. Nickel, M., Rosasco, L., Poggio, T.A.: Holographic embeddings of knowledge graphs. In: AAAI, pp. 1955–1961. AAAI Press (2016)
16. Nickel, M., Tresp, V., Kriegel, H.: A three-way model for collective learning on multi-relational data. In: ICML, pp. 809–816. Omnipress (2011)
17. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 10843, pp. 593–607. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93417-4_38
18. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**, 1929–1958 (2014)
19. Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., Gamon, M.: Representing text for joint embedding of text and knowledge bases. In: EMNLP, pp. 1499–1509. The Association for Computational Linguistics (2015)
20. Trouillon, T., Nickel, M.: Complex and holographic embeddings of knowledge graphs: a comparison. In: StarAI, vol. abs/1707.01475 (2017)
21. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: ICML. JMLR Workshop and Conference Proceedings, vol. 48, pp. 2071–2080. JMLR.org (2016)
22. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: a survey of approaches and applications. IEEE Trans. Knowl. Data Eng. **29**(12), 2724–2743 (2017)
23. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Learning multi-relational semantics using neural-embedding models. In: ICLR (2015)