Learning to Interact: An Adaptive Interaction Framework for Knowledge Graph Embeddings

Chandrahas[1] **Nilesh Agrawal**[2*] **Partha Talukdar**[1]
[1]Indian Institute of Science, Bangalore, [2]Cohesity, Bangalore
(chandrahas, anilesh, ppt)@iisc.ac.in

Abstract

Knowledge Graph (KG) Embedding methods have been widely studied in the past few years and many methods have been proposed. These methods represent entities and relations in the KG as vectors in a vector space, trained to distinguish correct edges from the incorrect ones. For this distinction, simple functions of vectors' dimensions, called interactions, are used. These interactions are used to calculate the candidate tail entity vector which is matched against all entities in the KG. However, for most of the existing methods, these interactions are fixed and manually specified. In this work, we propose an automated framework for discovering the interactions while training the KG Embeddings. The proposed method learns relevant interactions along with other parameters during training, allowing it to adapt to different datasets. Many of the existing methods can be seen as special cases of the proposed framework. We demonstrate the effectiveness of the proposed method on link prediction task by extensive experiments on multiple benchmark datasets.

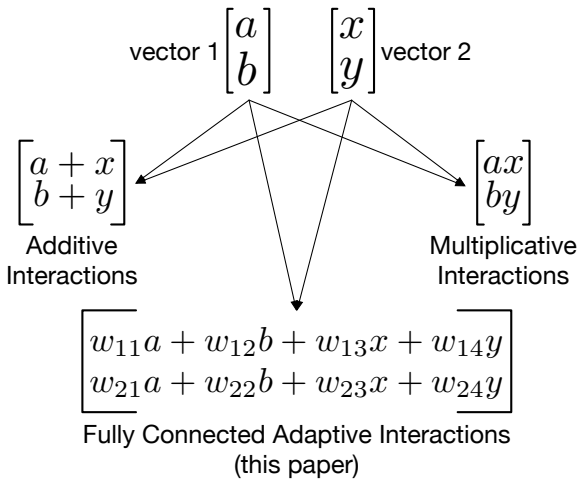


Figure 1: Some examples of interactions between two vectors. While additive and multiplicative interactions depend only on the input vectors, the fully connected (FC) interactions have trainable weights, allowing it to adapt to different datasets. In this example, additive interaction $a + x$ can be achived by FC interaction by setting $w_{11} = w_{13} = 1$ and $w_{12} = w_{14} = 0$. Similarly, the multiplicative interaction ax can be achieved by setting $w_{11} = 0.5x$ and $w_{13} = 0.5a$ and other weights as zero.

1 Introduction

Knowledge Graphs (KGs) such as NELL (Mitchell et al., 2015), Freebase (Bollacker et al., 2008), YAGO (Suchanek et al., 2007), etc. have been very popular in supporting many AI applications like Web Search Query Recommendation (Huang et al., 2016), Question Answering (Yao and Van Durme, 2014), Visual Question Answering (Shah et al., 2019) etc. KGs are multi-relational graphs containing entities as nodes and typed relations between entity pairs as edges. These graphs store real-world facts such as *(Lionel Messi, plays-for-team, Argentina National Football Team)* as edges, also called triples. In spite of their popularity, they suffer from incompleteness (Dong et al., 2014), and it becomes important to predict the missing edges in the graph. The task of predicting missing edges in a KG is called link prediction.

Knowledge Graph Embedding (KGE) methods have been a popular approach for the link prediction task. Most of these methods learn vectorial representations for entities and relations in the KG. A score function is then used to distinguish correct triples from the incorrect ones. Given a triple of the form (h, r, t) where h, r and t are the head entity, relation, and the tail entity, a score function assigns a real-valued score to the triple. These score functions depend upon the interactions of dimensions

*Work done while at the Indian Institute of Science, Bangalore.

Proceedings of the 17th International Conference on Natural Language Processing, pages 60–69
Patna, India, December 18 - 21, 2020.

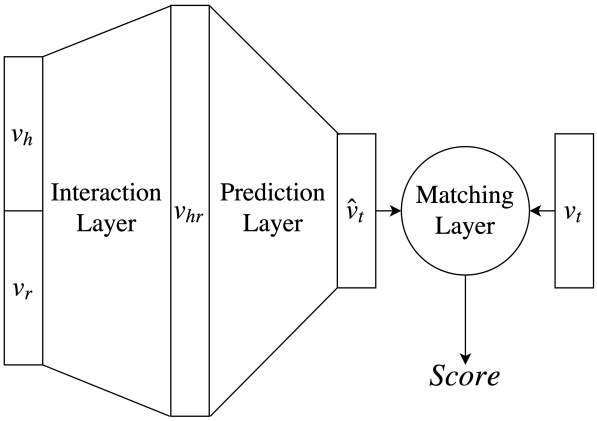


Figure 2: The block diagram of the Adaptive Interaction Framework. The interaction layer extracts interactions v_{hr} from head entity v_h and relation vector v_r. The prediction layer calculates a candidate tail entity vector $\hat{v}_t$ which is then matched with existing tail entity vector v_t using the matching layer to produce a real valued score.

of vectors for h and r. Some examples of interactions are given in Figure 1. TransE (Bordes et al., 2013) uses the additive interactions while DistMult (Yang et al., 2014) uses the multiplicative interactions. However, these interactions are fixed for a given method and are not learnable. This restricts models' capability to weigh entities and relations differently, and hence, from adapting to different datasets. For instance, relations in Freebase like *place_of_birth* give much more information about head and tail entities compared to relations like *_similar_to*, *_hypernym* in WordNet. Thus, learning these interactions while training can enable the model to adapt to different datasets.

We address this issue in this paper and propose a novel adaptive framework that allows learning these interactions directly from the data during training. The proposed framework is capable of weighing entities and relations differently by using a fully connected interaction layer. It allows the proposed method to adapt to different datasets by learning dataset-specific interactions. By extensive experiments on multiple benchmark datasets, we show the effectiveness of the proposed method on the link prediction task. We also demonstrate that the proposed method assigns different weights to entities and relations by learning dataset-specific interactions.

In summary, we make the following contributions:

- We propose an adaptive interaction framework that can discover relevant interactions of embeddings from data. We show that many of the existing methods can be seen as special cases of the proposed framework.
- Based on this framework, we propose two new models FCE and FCConvE which outperform the baseline models on link prediction task across commonly used benchmark datasets.
- We also present a method to analyse the fully connected interactions and use it to compare the interactions learned by FCConvE for different datasets.

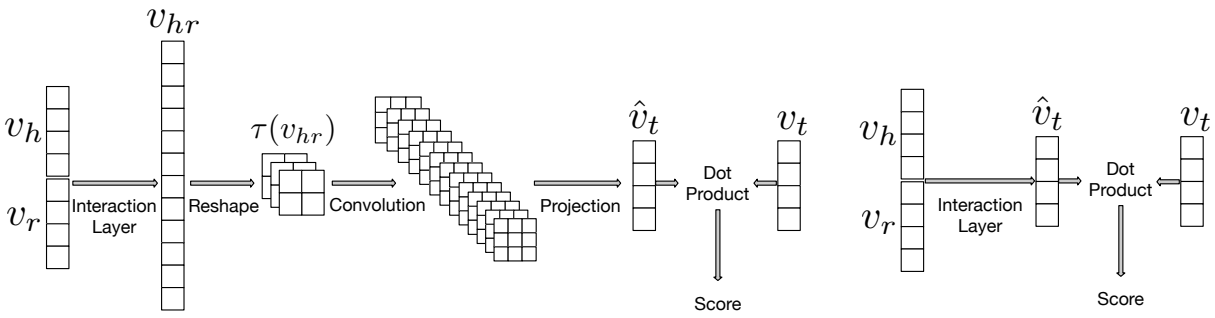
Notations: A Knowledge Graph is represented by $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ where $\mathcal{E}$ is the set of entities, $\mathcal{R}$ is the set of relations and $\mathcal{T} \subset \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is the set of triples stored in the graph. Most of the KG embedding methods learn vectors $v_e \in \mathbb{R}^{d_e}$ for $e \in \mathcal{E}$, and $v_r \in \mathbb{R}^{d_r}$ for $r \in \mathcal{R}$. Some methods also learn projection matrices $M_r \in \mathbb{R}^{d_r \times d_e}$ for relations. The correctness of a triple is evaluated using a model specific score function $score : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \rightarrow \mathbb{R}$. For learning the embeddings, a loss function $\mathcal{L}(\mathcal{T}, \mathcal{T}'; \theta)$, defined over a set of positive triples $\mathcal{T}$, a set of (sampled) negative triples $\mathcal{T}'$, and the parameters θ is optimized. I_d denotes the $d \times d$ identity matrix while 0_d denotes the $d \times d$ zero matrix. $diag(v)$ denotes a diagonal matrix created from vector v. All vectors are assumed to be column vectors including the concatenation of vectors $[v_1; v_2; \ldots v_k]$. In case of matrix, $[M_1; M_2]$ denotes the block matrix consisting of blocks M_1 and M_2.

2 Related Work

The problem of learning KG Embeddings for link prediction has been very popular in the last few years. Based on the score function, these methods can be broadly grouped into three categories, namely Additive, Multiplicative and Neural models.

2.1 Additive Models

This is the class of methods where the vectors interact via additive operations after an optional projection operation. One of the simple and popular additive models is TransE (Bordes et al., 2013) where the entity and relation vectors lie in the same vector space. The relation vector acts as a translation from the head entity vector to the tail entity vector.



(a) FCConvE (b) FCE

Figure 3: Architecture diagrams for FCConvE (left) and FCE (right). Please refer to Section 3.3 and Section 3.4 for more details.

SE (Bordes et al., 2011) is another model that uses relation specific similarity between head and tail entity vectors. Following the ideas of translation and projection, many different methods have been developed. These include TransH (Wang et al., 2014), TransR (Lin et al., 2015), STransE (Nguyen et al., 2016), ITransF (Xie et al., 2017), etc. These methods can only extract a restricted set of interactions from the vectors.

2.2 Multiplicative Models

In this class of methods, the vectors interact via a multiplicative operation. One of the initial models in this category is RESCAL (Nickel et al., 2011), which is based on tensor factorization. It models entities as vectors while relations as matrices. DistMult (Yang et al., 2014) is a special case of RESCAL where the relations matrices are restricted to be diagonal. However, DistMult score function is symmetric and hence it can not handle asymmetric relations. To alleviate this issue, HolE (Nickel et al., 2016) was proposed which uses circular correlation operation between head and tail entity vectors. ComplEx (Trouillon et al., 2016) addresses the same issue by modelling vectors in the complex domain. The asymmetry of complex dot product allows it to handle symmetric, asymmetric as well as anti-symmetric relations with the same score function. SimplE (Kazemi and Poole, 2018) is a more recent model based on tensor factorization which can express all types of relations. RotatE (Sun et al., 2019) is another model which uses complex vectors for representation. It models relation vectors as rotations from head to tail entity vector and then uses L1-norm based distance as score function. Similar to additive models, multiplicative models are also restricted in terms of the interactions they can extract.

2.3 Neural Models

There are many models that use various neural network architectures for learning KG Embeddings. Some of these models are NTN (Socher et al., 2013), ER-MLP (Dong et al., 2014), CONV (Toutanova et al., 2015), ProjE (Shi and Weninger, 2017), R-GCN (Schlichtkrull et al., 2017), ConvE (Dettmers et al., 2018), R-MLP-2n (Ravishankar et al., 2017), KG-BERT (Yao et al., 2019), InteractE (Vashishth et al., 2020), etc. Unlike other methods, KG-BERT uses word embeddings for encoding entities and relations. Therefore, the interactions of entity and relation vectors are not directly clear. Among the rest of the models, ProjE can extract interactions only from entity or relation vectors but not both. ConvE can extract interactions from both, entity as well as relation vectors, but they are extracted using predefined permutations. InteractE exploits more sophisticated interactions using feature permutation, checkered reshaping, and circular convolution resulting in improved performance. However, these methods depend on a fixed set of interactions defined by the model and can not adaptively learn these interactions during training.

As described in the next section, the models proposed in this paper are neural models that can adaptively learn the interactions from the entity as well well as relation vectors using a fully connected interaction layer.

3 Proposed Method

3.1 The Adaptive Interaction Framework

As shown in Figure 2, it consists of the following three components

Type	Model	Score Function $\sigma(h,r,t)$	Interactions
Additive	SE (Bordes et al., 2011)	$-\left\|M_r^1 v_h - M_r^2 v_t\right\|_p$	Manual
	TransE (Bordes et al., 2013)	$-\left\|v_h + v_r - v_t\right\|_p$	Manual
	TransR (Lin et al., 2015)	$-\left\|M_r v_h + v_r - M_r v_t\right\|_p$	Manual
	STransE (Nguyen et al., 2016)	$-\left\|M_r^1 v_h + v_r - M_r^2 v_t\right\|_p$	Manual
Multiplicative	DistMult (Yang et al., 2014)	$\langle v_r, (v_h \odot v_t)\rangle$	Manual
	HolE (Nickel et al., 2016)	$\langle v_r, (v_h \star v_t)\rangle$	Manual
	ComplEx (Trouillon et al., 2016)	$\text{Real}(\langle v_r, (v_h \odot \bar{v_t})\rangle)$	Manual
	RotatE (Sun et al., 2019)	$-\left\|v_h \odot v_r - v_t\right\|_p$	Manual
Neural	ER-MLP (Dong et al., 2014)	$\sigma(\langle \beta, \sigma(A \times [v_h; v_r; v_t])\rangle)$	Automatic
	ConvE (Dettmers et al., 2018)	$\langle \sigma(\text{vec}(\sigma(\tau(v_{hr}) * \Omega))U), v_t\rangle$	Manual
Adaptive Interactions Models	FCE (This paper)	$\langle \sigma(W \times [v_h; v_r]), v_t\rangle$	Automatic
	FCConvE (This paper)	$\langle \sigma(\text{vec}(\sigma(\tau(\sigma(W \times [v_h; v_r])) * \Omega))U), v_t\rangle$	Automatic

Table 1: Summary of various Knowledge Graph (KG) embedding methods mentioned in the paper. Here v_h, v_r, v_t denote the head entity, relation and tail entity vectors respectively. M_r, M_r^1, M_r^2 represent the relation specific projection matrices. $\|\cdot\|_p$ denotes the L1-norm ($p = 1$) or L2-norm ($p = 2$). $\langle\cdot,\cdot\rangle$, $\odot$, $\star$ and $*$ represent the inner product, the Hadamard product, circular correlation and convolution operations respectively. A and β represent the first and second layer weight matrices in ER-MLP respectively. Ω represents the convolution filters while U represents the final projection matrix in ConvE. $\tau(v_{hr})$ is a reshaped permutation of $[v_h; v_r]$ with optional repetition. vec denotes vectorization step followed by an activation function σ. Please refer to Section 2 for more details.

3.1.1 Interaction Layer

This layer extracts the interactions between the head entity and relation vectors which are later used for predicting tail entities. The interactions are extracted using a fully-connected (FC) layer:

$$v_{hr} = \sigma(W_r \times [v_h; v_r]), \tag{1}$$

where σ is an activation function. In the general case, W_r is $\mathbb{R}^{d_i \times (d_e + d_r)}$ where d_i is the dimension of the interaction layer. A special case of interaction layer is when $W_r = W, \forall\, r \in \mathcal{R}$ which avoids over-parameterization. We use this special case for our experimentations.

3.1.2 Prediction Layer

This layer predicts a vector $\hat{v}_t$ for the candidate tail entity as

$$\hat{v}_t = g(v_{hr}), \tag{2}$$

where the function g depends on the method. Many of the methods like TransE and DistMult use identity function. For our experiments, we use ConvE's (Dettmers et al., 2018) architecture for this step.

3.1.3 Matching Layer

This is the final layer where the predicted tail entity is matched against a given tail entity producing the final score for the triple. The score function is given as

$$score(h, r, t) = f(\hat{v}_t, v_t). \tag{3}$$

Again, the matching function f is method dependent. We use the vector dot product for matching.

3.2 Existing models as special case

In this section, we demonstrate how the proposed framework generalizes many of the existing models. The score functions for these models can be found in Table 1.

TransE: In TransE, the entities and relations lie in the same $d_e = d_r = d$ dimensional space $\mathbb{R}^d$. The interaction matrix is $W_r = [I_d; I_d]$ with identity activation while the tail prediction function $\hat{v}_t = g(v_{hr}) = v_{hr}$ is the identity function. The matching function takes the form of a vector norm $-\left\|\hat{v}_t - v_t\right\|_p$.

STransE: STransE (Nguyen et al., 2016) generalizes many translation-based models like TransH (Wang et al., 2014) and TransR (Lin et al., 2015) by using two relation specific projection matrices, M_r^1 for head and M_r^2 for tail entities. The interaction matrix is $W_r = [M_r^1; I_{d_r}]$ while the activation and prediction functions are identity functions. The matching function takes the form of $-\left\|\hat{v}_t - M_r^2 v_t\right\|_p$.

DistMult: For DistMult (Yang et al., 2014), the interaction matrix is $W_r = [diag(v_r), 0_d]$ while the activation and prediction functions are identity functions. Vector dot product between $\hat{v}_t$ and v_t is used as matching function.

ConvE: ConvE (Dettmers et al., 2018) is a recent

	FB15K	FB15K-237	WN18	WN18RR
#Entities	14,941	14,541	40,943	40,943
#Relations	1,345	237	18	11
#Train	483,142	272,115	141,441	86,835
#Validation	50,000	17,535	5,000	3,034
#Test	59,071	20,466	5,000	3,134

Table 2: Details of the datasets used in our experiments

model which uses a convolution network for extracting features from the interactions and then predicting the tail entity. Here, the interaction matrix is a fixed and manually specified permutation matrix, used for dimension shuffling in ConvE. Please note that the interaction matrix is shared among all relations. The prediction function is the 2D convolution network applied on the interactions vector v_{hr} while dot product between $\hat{v}_t$ and v_t is used for matching.

Based on the proposed framework, we present two new models FCConvE and FCE in the following sections. Their architecture can be found in Figure 3.

3.3 FCConvE

One issue with the existing methods is that the interactions are either fixed or it requires to be specified manually. For example, in the case of ConvE the permutations are specified by the user. This leaves the choice of interactions to the user and also does not specify which ones are better than others. Also, the performance on the link prediction task varies with different choices of permutations in ConvE. Hence it will be useful if the optimal interactions can be learned directly from the dataset.

FCConvE addresses this exact issue and allows the model to learn the appropriate interactions while training. It achieves this by using interaction matrix $W_r = W, \forall r \in \mathcal{R}$ as model parameter. Since many of the existing interactions can be achieved using different W matrices, the proposed model is capable of choosing the optimal interactions by itself. The score function for FCConvE can be given as follows:

$$score(h, r, t) = \langle \sigma(\text{vec}(\sigma(\tau(v_{hr}) * \Omega))U), v_t \rangle, \tag{4}$$

where v_{hr} is given by (1). As compared to ConvE, FCConvE contains approximately $\mathcal{O}(d_i \times (2d_e + d_r))$ more parameters due to the interaction and projection layers weights.

3.4 FCE

We experiment with a modification of DistMult named DistMult-BCE which uses Binary Cross Entropy (BCE) loss instead of margin-based ranking loss used in (Yang et al., 2014). We also introduce its adaptive interactions variant FCE (Fully Connected Embedding) which uses an FC layer as in (1) for interactions instead of Hadamard product between v_h and v_r. The score function for FCE can be given as follows:

$$score(h, r, t) = \langle \sigma(W \times [v_h; v_r]), v_t \rangle. \tag{5}$$

We use ReLU for the activation function σ. Please note that, similar to DistMult, entity and relation vectors lie in same $d_e = d_r = d$ dimensional space $\mathbb{R}^d$. The interaction vector v_{hr} also lies in the same space $\mathbb{R}^d$ (i.e. $d_i = d$). Unlike DistMult, the interaction matrix $W \in \mathbb{R}^{d \times d}$ is shared across relations. In terms of model size, FCE contains $d_i \times (d_e + d_r) = 2d^2$ more parameters as compared to DistMult.

3.5 Analysing Interactions using NAIW

In this section, we introduce a novel method to analyse fully connected interactions. As mentioned in previous sections, the interaction weight matrix W in FCConvE as well as FCE is shared among all relations i.e. $W_r = W, \forall r \in \mathcal{R}$. This interaction weight matrix can be split into two parts, $W = [W^{\mathcal{E}}; W^{\mathcal{R}}], W^{\mathcal{E}} \in \mathbb{R}^{d_i \times d_e}, W^{\mathcal{R}} \in \mathbb{R}^{d_i \times d_r}$ corresponding to the head entity and the relation vectors respectively. The equation for the interaction layer can be re-written as follows:

$$v_{hr} = \sigma(W^{\mathcal{E}} v_h + W^{\mathcal{R}} v_r). \tag{6}$$

The values in $W^{\mathcal{E}}$ and $W^{\mathcal{R}}$ represent the importance of various dimensions of the head entity and relation vectors. We use the absolute values of these weights for comparing the importance of entity and relation. Specifically, we use the *Normalized Absolute Interaction Weights (NAIW)* as defined below for comparing the weights corresponding to entities and relations.

$$\begin{aligned} V^{\mathcal{E}} &= \sum_{j=1}^{d_e} \text{AbsoluteValue}(W^{\mathcal{E}}[:, j]), \\ \text{NAIW}^{\mathcal{E}} &= \frac{V^{\mathcal{E}}}{\max(V^{\mathcal{E}})}, \end{aligned} \tag{7}$$

where $V^{\mathcal{E}} \in \mathbb{R}^{d_i}$ is a vector containing sum of absolute interaction weights across dimensions of

Model	FB15K-237					WN18RR				
	MR↓	MRR↑	Hits↑			MR↓	MRR↑	Hits↑		
			@1	@3	@10			@1	@3	@10
DistMult (Yang et al., 2014)	254	24.1	15.5	26.3	41.9	5110	43.0	39.0	44.0	49.0
ComplEx (Trouillon et al., 2016)	339	24.7	15.8	27.5	42.8	5261	44.0	41.0	46.0	51.0
R-GCN (Schlichtkrull et al., 2017)	-	24.8	15.3	25.8	41.7	-	-	-	-	-
ConvE (Dettmers et al., 2018)	**244**	32.5	23.7	35.6	50.1	4187	43.0	40.0	44.0	52.0
DistMult-BCE	318	30.1	21.6	33.0	46.9	7037	41.0	38.6	41.6	46.0
FCE	331	30.6	21.7	33.7	48.6	4732	41.3	38.2	42.2	47.5
FCConvE	255	**35.5**	**26.4**	**39.1**	**54.0**	**4103**	**46.1**	**42.8**	**47.7**	**52.7**

Model	FB15K					WN18				
	MR↓	MRR↑	Hits↑			MR↓	MRR↑	Hits↑		
			@1	@3	@10			@1	@3	@10
TransE (Bordes et al., 2013)	-	46.3	29.7	57.8	74.9	-	49.5	11.3	88.8	94.3
DistMult (Yang et al., 2014)	97	65.4	54.6	73.3	82.4	902	82.2	72.8	91.4	93.6
ComplEx (Trouillon et al., 2016)	-	69.2	59.9	75.9	84.0	-	94.1	93.6	93.6	94.7
R-GCN (Schlichtkrull et al., 2017)	-	69.6	60.1	76.0	84.2	-	81.4	69.7	92.9	**96.4**
ConvE (Dettmers et al., 2018)	**51**	65.7	55.8	72.3	83.1	**374**	94.3	93.5	94.6	95.6
DistMult-BCE	115	73.3	66.8	77.8	84.9	671	83.9	74.8	92.6	94.7
FCE	108	**74.6**	**67.8**	**79.5**	**86.1**	516	94.2	93.6	94.5	95.2
FCConvE	67	71.7	63.4	77.3	85.6	440	**94.8**	**94.3**	**95.1**	95.5

Table 3: Link prediction results on benchmark datasets. Here ↑ indicates higher values are better while ↓ indicates lower values are better. The adaptive interaction versions of the models FCConvE and FCE outperform the corresponding baseline models ConvE and DistMult-BCE in all the datasets. They also outperform other methods across all datasets. Results for the baseline models except TransE were taken from (Dettmers et al., 2018). For TransE, we have taken the results from (Nickel et al., 2016). We have also included results for a modification of DistMult called DistMult-BCE. Please refer to Section 4.2 for more details.

head entity. It denotes the importance of the head entity for each unit in the interaction layer. We normalize this vector such that the values lie in $[0, 1]$ range. This allows us to compare this value across datasets. Similarly, we can calculate $V^{\mathcal{R}}$ and NAIW$^{\mathcal{R}}$ which denote the importance of relation for units in the interaction layer.

Each value in the NAIW vector represents the importance of entities (for NAIW$^{\mathcal{E}}$) or relations (for NAIW$^{\mathcal{R}}$) for the link prediction task. Thus, comparing these values helps us understand the relative importance of entities and relations for different datasets. Since a comparison of individual interaction units may be inconclusive, we compare their distributions. We estimate the distributions[1] of NAIW$^{\mathcal{E}}$ and NAIW$^{\mathcal{R}}$ and compare them across multiple datasets. These distributions allow us to compare the importance of the entity and relation for the link prediction task.

[1]We use *gaussian_kde* function from SciPy library for estimating distributions.

4 Experiment Results

In this section, we evaluate the proposed method on the link prediction task and compare it against the baselines. The details of the datasets used for evaluation are given in Table 2. We provide the implementation details followed by the results in the following sections.

4.1 Implementation details

In our experiments, we use 200-dimension embeddings for both entity as well as relations. For selecting other hyper-parameters, we use cross-validation using the MRR on validation split of the data. Similar to ConvE, we use dropouts and batch normalization at input, convolution and final projection layer. The corresponding dropout probabilities are selected from [0.1, 0.2, 0.3], [0.2, 0.3, 0.4] and [0.4, 0.5, 0.6] respectively. For the interaction layer, we use 5000 dimensions reshaped to $25 \times 10 \times 20$ before applying convolution. Unlike ConvE, FCConvE uses depthwise group convolution. The

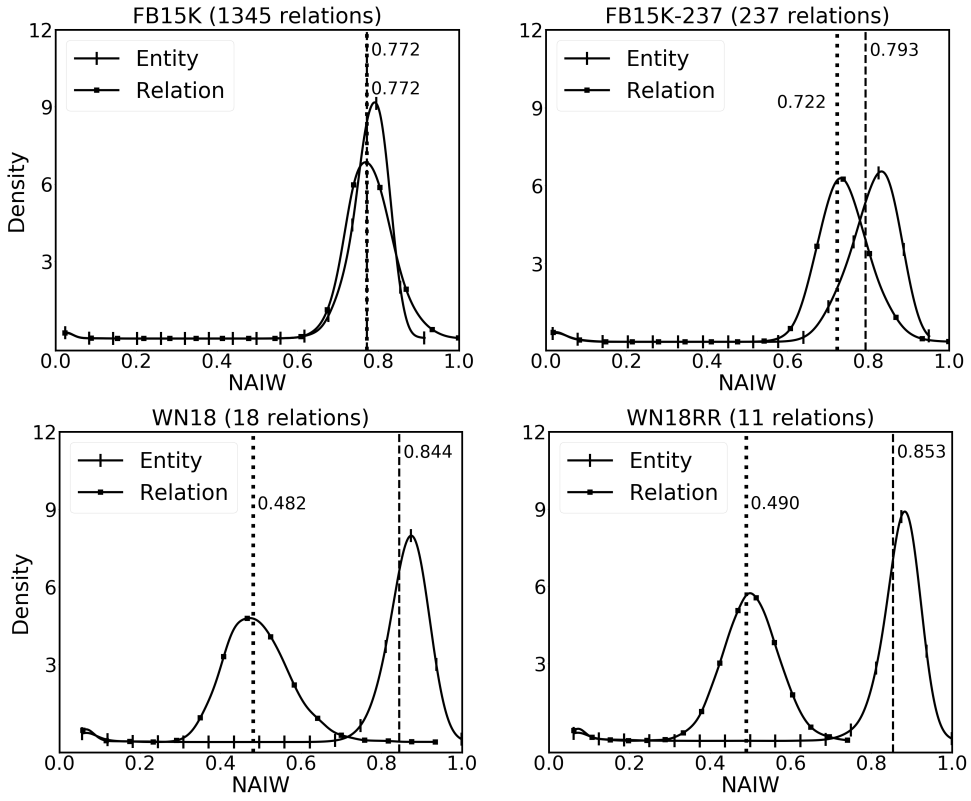


Figure 4: Distributions of the Normalized Absolute Interaction Weights for entities and relations learned by FCConvE on different datasets. The means of these distributions are shown as a dashed (for Entity) or dotted (for Relation) vertical lines along with their values. The datasets are arranged according to decreasing order of number of relations from left to right. As we can see, for lower number of relations, the difference in weights for entities and relations are much higher. Please refer to Section 4.3 for more details.

filter size for convolution is selected from [3×3, 5×5]. For optimization, we use Adam optimizer with an initial learning rate of 0.001. We use early stopping using MRR on validation split with the maximum number of epochs set to 100. The best model is then run for 1000 epochs[2] and final performance is reported. We use 500 negative samples per correct triple. For head entity prediction, we follow ConvE and use reversed relations during training as well as evaluation. The embeddings are trained using binary cross-entropy loss.

4.2 Link Prediction

Given a test or validation triple, we score the head entity and relation against all entities and report the rank of the correct tail entity. A similar strategy is used for head entity prediction except that we use reverse relation vectors. The model's performance is evaluated on both head as well as tail entity prediction and the average performance is reported. Similar to previous work, we use filtered setting, i.e., we exclude all triples appearing in train, test or validation split while ranking. We report Mean Reciprocal Rank (MRR), Mean Rank (MR) and Hits@k for k=1, 3, 10 on test split. The results can be found in Table 3.

For comparison, we use a few representative baselines from each category of the models. Specifically, we use TransE for additive models, DistMult and ComplEx for multiplicative models, and R-GCN and ConvE for neural models. Please note that our goal is to compare a model with its adaptive interaction version, instead of comparing all available models.

We observe from the results that the proposed adaptive interaction versions of the models outperform the corresponding baseline models in all the datasets. FCConvE significantly outperforms ConvE in all the datasets except WN18 where the performance is comparable. Similarly, FCE significantly outperforms DistMult-BCE in WN18 while showing marginal improvements is other datasets. Also, they outperform other methods on the link prediction task across all the datasets suggesting

[2]We pick the model from the epoch with the best validation split MRR, and it need not be the final epoch.

that the proposed approch is able to adapt to different datasets resulting in performance improvements.

It should be noted that ConvE would struggle to differentially weigh entity and relation vector dimensions due to the sharing of convolution filters. Adding an FC interaction layer allows it to prioritize between entity and relation vectors resulting in better performance of FCConvE. We also observe that the DistMult-BCE model significantly outperforms the DistMult model in FB15K and FB15K-237 which suggests the BCE loss with multiple negative samples improves performance.

Among the metrics used in Table 3, MR is more sensitive to outliers (i.e., large values of ranks) than others. We observe that the proposed approaches achieve improvements in MRR and Hits@k, but not MR for many datasets. It suggests that the proposed methods are effective in bringing more cases into the high-rank region, which could be a desirable property in many applications.

4.3 Interactions Analysis

In this section, we analyse the interactions learned by FCConvE on different datasets. We use the distributions of NAIW$^{\mathcal{E}}$ and NAIW$^{\mathcal{R}}$, as defined in Section 3.5 and compare them. The results are shown in Figure 4.

As we can see from Figure 4, the distributions of NAIW$^{\mathcal{E}}$ and NAIW$^{\mathcal{R}}$ varies across different datasets. Furthermore, we make the following observations.

- The difference between the means of NAIW$^{\mathcal{E}}$ and NAIW$^{\mathcal{R}}$ increases with decreasing number of relations. Among the datasets used, FB15K has the most number of relations (i.e. 1,345) while WN18RR has the least number of relations (i.e. 11). This number is correlated with the difference of the means of NAIW$^{\mathcal{E}}$ and NAIW$^{\mathcal{R}}$ with WN18RR having the highest difference, while FB15K having the lowest difference. This suggests that when a dataset has a small number of relations, entities have more distinguishing capability than the relations.

- The relations in Freebase datasets (i.e. FB15K and FB15K-237) have more distinguishing capability than relations in WordNet datasets (i.e. WN18 and WN18RR). For example, relations like *place_of_birth* in Freebase restricts candidate entity types for head and tail entities. On the other hand, relations in Wordnet (e.g. *_similar_to*, *_hypernym*) are not very specific to some type of entities. This behavior is reflected in the distributions of NAIW$^{\mathcal{E}}$ and NAIW$^{\mathcal{R}}$ with relations getting more weights in Freebase datasets as compared to WordNet datasets.

4.4 Effect of various interactions on ConvE

To further understand the advantages of adaptive interactions, we compare its performance with various fixed interactions (as used in (Vashishth et al., 2020)). As mentioned in Section 3.2, ConvE can use a permutation matrix for generating interactions. For demonstration, let's assume a head entity vector $v_h = [v_h^1, v_h^2, v_h^3, v_h^4, v_h^5, v_h^6]$ and a relation vector $v_r = [v_r^1, v_r^2, v_r^3, v_r^4, v_r^5, v_r^6]$. These vectors are concatenated, permuted and then reshaped into a 4×3 matrix before passing it to the convolution layer. As shown in Figure 5, the following are some of the candidate permutations.

Plain: A plain concatenation and reshaping of the vectors.

Alternate Rows: Alternate rows of head entity and relation vectors dimensions.

Alternate: Strictly alternating dimensions of head entity and relation vectors.

$$\begin{bmatrix} v_h^1 & v_h^2 & v_h^3 \\ v_h^4 & v_h^5 & v_h^6 \\ v_r^1 & v_r^2 & v_r^3 \\ v_r^4 & v_r^5 & v_r^6 \end{bmatrix} \quad \begin{bmatrix} v_h^1 & v_h^2 & v_h^3 \\ v_r^1 & v_r^2 & v_r^3 \\ v_h^4 & v_h^5 & v_h^6 \\ v_r^4 & v_r^5 & v_r^6 \end{bmatrix} \quad \begin{bmatrix} v_h^1 & v_r^1 & v_h^2 \\ v_r^2 & v_h^3 & v_r^3 \\ v_h^4 & v_r^4 & v_h^5 \\ v_r^5 & v_h^6 & v_r^6 \end{bmatrix}$$

(a) Plain (b) Alternate Rows (c) Alternate

Figure 5: Some example permutations of two 6-dimensional vectors v_h and v_r.

As we can see, *Plain* method allows entity-relation interactions only at the boundary region while *Alternate Rows* and *Alternate* allow deeper interactions.

We run the best hyper-parameters settings of ConvE with these three permutations on all the datasets and compare the MRR of the link prediction task. As seen from the results in Table 4, the *Alternate Rows* and *Alternate* permutation schemes achieve better results compared to *Plain* permutation scheme. However, since FCConvE can learn

Permutation	FB15K	FB15K-237	WN18	WN18RR
Plain	63.2	32.7	94.3	43.2
Alternate Rows	63.6	33.3	94.8	44.3
Alternate	63.9	33.3	94.8	44.4
FCConvE	71.7	35.5	94.8	46.1

Table 4: The effect of various permutation schemes on the performance of ConvE. We report the MRR in link prediction task across various datasets. As we can see, the performance of ConvE is dependent on the choice of permutation scheme and using Alternate or Alternate Rows permutation improves the performance of ConvE. Please refer to Section 4.4 for more details.

the interactions while training, it achieves better or comparable MRR on all the datasets.

5 Conclusions and Future Work

We presented an adaptive interaction framework for learning KG Embeddings and proposed two new models based on the framework. We demonstrated that the proposed models are capable of learning relevant interactions across different datasets. We also demonstrated how some of the existing KG Embedding models can be seen as special cases of the proposed framework.

In the future, we would like to further analyze the interaction layer and its correlation with more dataset properties.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work is supported by the Ministry of Human Resources Development (Government of India).

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. 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.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pages 1811–1818.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. 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.

Zhipeng Huang, Bogdan Cautis, Reynold Cheng, and Yudian Zheng. 2016. Kb-enabled query recommendation for long-tail queries. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 2107–2112, New York, NY, USA. ACM.

Seyed Mehran Kazemi and David Poole. 2018. Simple embedding for link prediction in knowledge graphs. In *Advances in Neural Information Processing Systems*, pages 4284–4295.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2015. Never-ending learning. In *Proceedings of AAAI*.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016. STransE: a novel embedding model of entities and relationships in knowledge bases. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 460–466, San Diego, California. Association for Computational Linguistics.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. 2016. Holographic embeddings of knowledge graphs. In *AAAI*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*, volume 11, pages 809–816.

Srinivas Ravishankar, Chandrahas, and Partha Pratim Talukdar. 2017. Revisiting simple neural networks for learning representations of knowledge graphs. *6th Workshop on Automated Knowledge Base Construction (AKBC) at NIPS 2017*.

M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling. 2017. Modeling Relational Data with Graph Convolutional Networks. *ArXiv e-prints*.

Sanket Shah, Anand Mishra, Naganand Yadati, and Partha Pratim Talukdar. 2019. Kvqa: Knowledge-aware visual question answering. In *AAAI*.

Baoxu Shi and Tim Weninger. 2017. Proje: Embedding projection for knowledge graph completion. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW*.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1499–1509, Lisbon, Portugal. Association for Computational Linguistics.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, Nilesh Agrawal, and Partha Talukdar. 2020. Interacte: Improving convolution-based knowledge graph embeddings by increasing feature interactions. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, pages 3009–3016. AAAI Press.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119. Citeseer.

Qizhe Xie, Xuezhe Ma, Zihang Dai, and Eduard Hovy. 2017. An interpretable knowledge transfer model for knowledge base completion. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 950–962, Vancouver, Canada. Association for Computational Linguistics.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Kg-bert: Bert for knowledge graph completion. *ArXiv*, abs/1909.03193.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with Freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 956–966, Baltimore, Maryland. Association for Computational Linguistics.