



Contextualise Entities and Relations: An Interaction Method for Knowledge Graph Completion

Kai Chen¹, Ye Wang^{1,2(✉)}, Yitong Li³, Aiping Li¹, and Xiaojuan Zhao¹

¹ National University of Defense Technology, Changsha 410000, China
{chenkai.,ye.wang,liaiping}@nudt.edu.cn

² School of Computer Science and Technology, Harbin Institute of Technology,
Shenzhen 518000, China

³ Huawei Device Co., Ltd., Shenzhen 518000, China
liyitong3@huawei.com

Abstract. The incompleteness of Knowledge Graph (KG) stimulates substantial research on knowledge graph completion, however, current state-of-the-art embedding based methods represent entities and relations in a semantic-separated manner, overlooking the interacted semantics between them. In this paper, we introduce a novel entity-relation interaction mechanism, which learns contextualised entity and relation representations with each other. We feature entity interaction embeddings by adopting a translation distance based method which projects entities into a relation-interacted semantic space, and we augment relation embeddings using a bi-linear projection. Built upon our interaction mechanism, we experiment our idea using two decoders, namely a simple Feed-forward based Interaction Model (FIM) and a Convolutional network based Interaction Model (CIM). Through extensive experiments conducted on three benchmark datasets, we demonstrate the advantages of our interaction mechanism, both of them achieving state-of-the-art performance consistently.

Keywords: Knowledge graph completion · Link prediction · Knowledge inference

1 Introduction

Extensive applications [14, 25] of real-world employ knowledge graphs, consisting of abundant facts [1, 2, 9]. Specifically, each fact is denoted by a triple, (h, r, t) , representing a head entity h has relation r with a tail entity t . In real KG applications, annotated facts are sparse and many facts remain unrevealed, thus requiring to be completed or inferred using the existing facts, i.e., Knowledge Graph Completion (KGC) [8, 17, 23]. One of the most common KGC tasks is link

This work is supported by the National Natural Science Foundation of China (No. 61732004, 61732022).

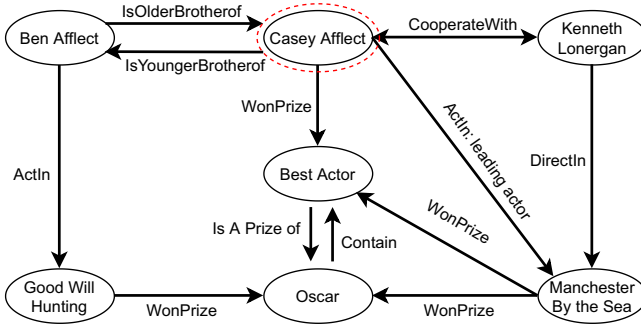


Fig. 1. An example of a knowledge graph. The nodes represent entities while the edges represent relations. The dashed red circle represents the entity to be predicted in a given incomplete triple. (Color figure online)

prediction that infers a missing entity in an incomplete triple, e.g. $(h, r, ?)$, or vice versa $(?, r, t)$.

Generally, tackling link prediction requires KGC model understanding the semantics of entities and relations, as well as the structure information of existing graph. For example, in Fig. 1, when inferring the tail entity in the incomplete fact $(Ben\ Affleck, IsOlderBrotherof, ?)$, one can imply that the tail entity should be a person rather than other type of entity given the semantic information of relation $IsYoungerBrotherof$, i.e. either *Casey Affleck* or *Kenneth Lon* in the current KG. Such semantics can usually be discovered by existing SOTA KGC methods [8, 16], however, this is not enough to distinguish these two candidates. In fact, *Casey Affleck* is “younger than” a person, given the semantics of the relation $IsYoungerBrotherof$, which makes *Casey Affleck* more likely to be the candidate. That is the semantics of the relation, as the contextual information of the entity, can provide sufficient information for learning the entity representations, while it is often omitted by the existing embedding methods, e.g. translation-based methods. On the other hand, when learning relation representations, the semantic of context entity can also be helpful. Furthermore, semantics of the mutual learning procedure will be aggregated. The idea of contextualised representations have been successfully applied to NLP applications [5, 15], however not well-studied for KGC. Therefore, in this paper, we test the hypothesis that learning contextualised representations of entity and relation can help KGC models better utilise and understand the semantics of surrounding entities and relations.

Recently, [13] considers a intuitive tensor factorization model that describes the semantic relevance of entities via a bi-linear transformation based on the entity relationship. And [22] relaxes the bi-linear transformation with a diagonal matrix. Inspired by their approaches of bi-linear inter-relation, to capture a contextual representation, we extend the idea of bi-linear interaction to entities and relations, where entity and relation are not only represented by their original embeddings but also contextualised with each other within the fact triple. And finally, the model is capable of learning contextualised representations for entity and relation.

Following the above inspirations, in this paper, we propose a novel interaction mechanism to better learn the contextualised entity and relation representations. Our interaction mechanism (1) adopts the idea of translational models [3, 8], i.e., we project the relation embedding into the entity space and integrate it with the entity embedding to form an interacted entity embedding, (2) takes advantages of bi-linear models for their semantic interaction [13, 20, 22], i.e., we take an element-wise *Hadamard product* over entity embedding, relation embedding and entity interactive embedding to form an interacted relation embedding. Based on these two interaction mechanisms, we explore two decoders: a state-of-the-art Convolutional Neural Net (CNN) and a feed-forward neural network, and we experiment with link prediction tasks across different benchmarks.

We summarize our main contributions as follows:

1. We propose a novel entity-relation interaction mechanism and experiment with a CNN and a feed-forward Interaction Models (CIM and FIM) based on the proposed interaction mechanism.
2. Our CIM achieves the SOTA performance on all of FB15k-237, WN18RR and YAGO3-10 datasets for link prediction task. Even without CNN, the simple FIM can achieve comparable performance with previous SOTA methods consistently, demonstrating the advantages of the proposed interaction mechanism.

2 Related Works

Translational Methods. Translational model is an important founder in KGC. Its main idea is based on translation distance, projecting both entities and relations into the same embedding space and utilizing a distance based reductive equation to reflect the translation constraint. Among them, TransE is the origin, which holds a translation distance for the relation between the corresponding head and tail entities for the triple (h, r, t) , that is, $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ [3]. Similarly, TransR also applies the translation distance but by projecting the embeddings of head and tail entities from entity space into relational space, and then uses translation distances for relations $\mathbf{h}_r + \mathbf{r} \approx \mathbf{t}_r$ [8].

Bi-linear Methods. Bi-linear models describe the semantic relevance of entities by a bi-linear transformation based on the relationship between entities, and it can effectively describe the coordination between entities. RESCAL expresses the relation by full rank matrix and defines the scoring function as $f_r(\mathbf{h}, \mathbf{t}) = \mathbf{h}^\top \mathbf{M}_r \mathbf{t}$ [13]. To avoid overfitting, Distmult relaxes the constraint on the relation matrix and replaces it with a diagonal matrix of relation by $f_r(\mathbf{h}, \mathbf{t}) = \mathbf{h}^\top \text{Diag}(\mathbf{M}_r) \mathbf{t}$ [22], however, only symmetric relations can be well studied.

Convolutional Network Methods. Beneficial from the powerful ability of feature extraction, many recent methods are developed upon CNN. ConvE introduces an CNN based model to KGC, which applies a 2D convolution network to extract

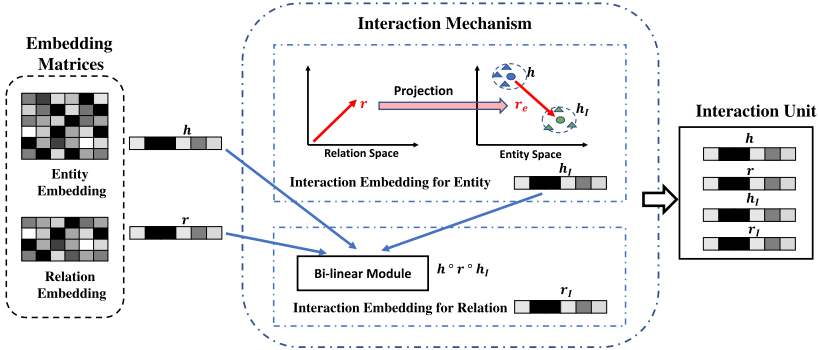


Fig. 2. An illustration of the proposed interaction mechanism. We take two different methods to get the interaction embedding for entity and the interaction embedding for relation, then we construct the interaction unit.

features [4]. Differently, ConvKB deploys 1D convolution filters rather than 2D filters and has been declared to be superior to ConvE [12]. However, the performance of ConvKB has been revealed inconsistent across different datasets and the evaluation protocol of ConKB is concerned [18]. To take advantages of the translation property, Shang et al. proposes ConvTransE [16]. CrossE makes attempts to simulate the crossover interactions between entities and relations [24] by considering the interaction as two parts: the interaction from relations to entities and vice versa. InteractE captures heterogeneous feature interactions through permutations over the embedding structure [21]. It realigns components from both entities and relations to form a so-called feature permutation followed by decoding procedure with ConvE which is quite different from our method that explores extra latent features from real interactions between entities and relations.

Graph Network Methods. Recently, more works [11, 16] aggregate inherent local graph neighbourhood to encode embeddings of entities. SACN [16] builds entity embedding matrix by a weighted GCN encoder but pays little attention on encoding relation embeddings before feeding them into the ConvTransE decoder. KBGAT [11] utilizes a simple transformation to the initial relation embedding for calculating the relative attention value for single triple to compute new embeddings of entities so that contextual semantics can hardly make contribution to relation embeddings beyond 2-hop neighbors. Moreover, the attention mechanism is questioned because of a data leakage problem [18].

3 Methodology

We apply an interaction method to learn contextualised entity and relation representations. Firstly, we define the interaction embeddings for both entity and

relation, which constitute an interaction unit as feature of the triple. The interaction unit integrates the superiorities of both translational models and bi-linear models by combining the representations of general embeddings and interaction embeddings. We then feed the interaction unit into the decoder to make a prediction. In this paper, we utilise two types of decoders, a feed-forward based model and a convolutional network based model to test our hypothesis.

3.1 Interaction Mechanism

The interaction mechanism is illustrated in Fig. 2. In this paper, we focus on the link prediction task, that is, given only one entity and the relation of a triple, we need to predict the other entity. We build interaction embeddings for both the given entity and the relation accordingly. For the convenience, we take the head entity h as the known entity and the tail entity t as the predicting entity in the following of this section.

Entity Interaction Embedding. Inspired by translation based models, we notice the strong interrelations among the basic elements of a triple: head entity, relation, and tail entity. For example, aiming at the task of predicting tail entity for a masked triple $(h, r, ?)$, we project the relation embedding \mathbf{r} into the entity space, and use a translation transformation over the embedding of head entity \mathbf{h} . The interaction embedding for entity is formulated as:

$$\mathbf{h}_I = \mathbf{h} + \mathbf{W}_e \mathbf{r}, \quad (1)$$

where \mathbf{W}_e denotes a trainable transferring matrix used to project the embedding of relation into the entity space. Intuitively, it also delivers information hidden in \mathbf{r} to \mathbf{h} .

We can describe Eq. 1 in terms of semantic spatial features. We divide the whole semantic space into entity space and relation space. When the head entity \mathbf{h} is taken as a central node and the edge corresponding to relation \mathbf{r} is considered, all neighbor nodes connected to the central via the edge should have features similar to \mathbf{h}_I in the entity space, as is illustrated in Fig. 2.

Relation Interaction Embedding. Different from entity interaction embedding, we take inspirations from bi-linear models to construct relation interaction embedding. Here, we utilize *Hadamard product*, an element-wise multiplication operator, to formulate the interaction embedding for relation:

$$\mathbf{r}_I = \mathbf{h} \circ \mathbf{r} \circ \mathbf{h}_I, \quad (2)$$

where \circ denotes *Hadamard product*.

One advantage of applying bi-linear function is that the interacted embedding is capable of representing the potential semantic information of entities and relations, which tends to obtain deep level interactions and inter-relationships between entities and relations. Note that all the embeddings above are vectorised in the same dimension $\mathbf{h}, \mathbf{h}_I, \mathbf{r}, \mathbf{r}_I \in \mathbb{R}^E$, where E is dimension of the embedding.

Interaction Unit. To enhance the features of latent interactions between entities and relations, we concatenate the two general embeddings \mathbf{h} and \mathbf{r} and the two interaction embeddings \mathbf{h}_I and \mathbf{r}_I to preserve more information. Finally, the interaction unit is formulated as:

$$\mathbf{U} = \text{concat}([\mathbf{h}, \mathbf{r}, \mathbf{h}_I, \mathbf{r}_I]) \in \mathbb{R}^{4 \times E}. \tag{3}$$

The interaction unit is further served as an input of the decoder models. By utilizing the interaction mechanism, we aim to make more information interaction of entity and relation directly from the embedding level, so that our models are expected to be more robust than having them separated in their own semantic spaces.

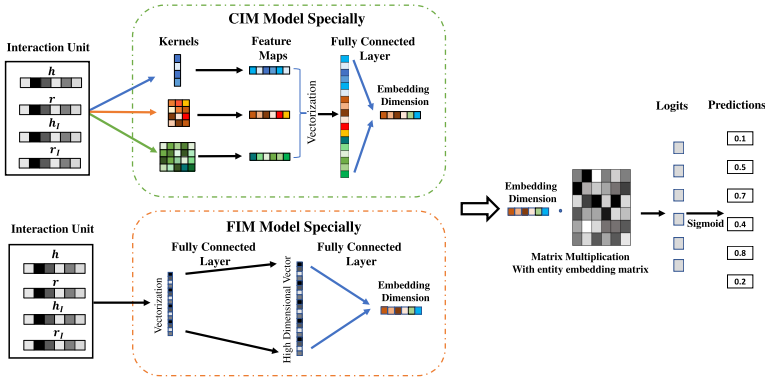


Fig. 3. An illustration of the two proposed methods, CIM and FIM. Both CIM and FIM take the proposed interaction unit as the input and utilize a convolutional neural network and a feed-forward neural network, respectively.

3.2 Interaction Embedding Decoders

Based on our interaction mechanism, we propose two methods to decode the tail entity.

Feed-Forward Neural Network Based Interacted Model (FIM). The first one is a simple feed-forward neural network based interacted model. The overall architecture of FIM is depicted in Fig. 3.

Taking the interaction unit \mathbf{U} as the input, the model firstly reshapes it into an 1d vector, and then puts it into a fully-connected (FC) layer to produce a hidden representation \mathbf{V}_H .

$$\hat{\mathbf{U}} = \text{vec}(\mathbf{U}) \tag{4}$$

$$\mathbf{V}_H = \text{FC}(\hat{\mathbf{U}}) = \text{relu}(\mathbf{W}_{fc} \hat{\mathbf{U}} + \mathbf{b}_{fc}), \tag{5}$$

where vec denotes the vectorization operator, FC denotes the fully-connected layer with the activation function relu , producing $\mathbf{V}_H \in \mathbb{R}^D$, where D is the dimension of the hidden layer.

Then we transform \mathbf{V}_H into a predicted embedding $\mathbf{V} \in \mathbb{R}^E$ parameterised by a transformation matrix $\mathbf{W}_F \in \mathbb{R}^{D \times E}$ and a *non-linear* function f :

$$\mathbf{V} = f(\mathbf{V}_H \mathbf{W}_F). \quad (6)$$

And \mathbf{V} is the acquired prediction vector for the target entity of the task.

Finally, we measure the similarity between the prediction embedding and the candidate tail entity embedding using *dot product*, formulated as $\mathbf{V} \cdot \mathbf{t}$, where \mathbf{t} denotes the embedding of a tail entity. And the overall scoring function of a triple is summarised as following:

$$f(\text{FC}(\text{vec}([\mathbf{h}, \mathbf{r}, \mathbf{h}_I, \mathbf{r}_I]))) \cdot \mathbf{t}. \quad (7)$$

For training, we calculate the scoring function for all entities and minimise the cross-entropy loss between the score logits (with a softmax) and the true label. For inference, we choose the entity with high score as the predicted entity.

CNN Based Interacted Model (CIM). We also propose a novel CNN based interacted model CIM. We take advantages of previous ConvTransE [16], and the main idea of our CIM is also illustrated in Fig. 3.

Based on the interaction unit \mathbf{U} , we take advantages of the convolutional neural network for its computation efficiency and powerful ability to learn features. We first apply the convolutional filters on \mathbf{U} and generate the output \mathbf{C} as follow:

$$\mathbf{C} = \text{conv}(\mathbf{U}), \quad (8)$$

where conv denotes a convolution operator. The convolution operator uses N_C channels, each applying a 1D convolutional filter with size $4 \times k$, where k is kernel width. Specifically, the c -th kernel is parameterised by ω_c (ω_c is trainable), and the convolution utilized here is as follows:

$$m_c(n) = \sum_{\tau=0}^{k-1} \sum_{j=0}^3 \omega_c(\tau, j) \hat{\mathbf{U}}_j(n + \tau), \quad (9)$$

where the $\hat{\mathbf{U}}$ over a vector denotes a padding version of the corresponding vector, and n indexes the entries in the output vector. And the output vector corresponding to the c -th kernel is formulated as $\mathbf{M}_c = [m_c(0), \dots, m_c(E-1)]$. Considering the channel size N_C , our output of the convolution \mathbf{C} is actually a combination of different \mathbf{M}_c , and $\mathbf{C} \in \mathbb{R}^{E \times N_C}$.

To predict the tail entity from a masked triple $(h, r, ?)$, we resize \mathbf{C} to a vector:

$$\mathbf{V} = f(\text{vec}(\mathbf{C}) \mathbf{W}_C), \quad (10)$$

where vec again denotes reshaping operator to reshape a matrix shaped as $\mathbb{R}^{E \times N_C}$ into a vector $\mathbf{V} \in \mathbb{R}^{EN_C}$, \mathbf{W}_C denotes the parameter for the linear transformation ($\mathbf{W}_C \in \mathbb{R}^{EN_C \times E}$), and f denotes a non-linear function.

Similar to FIM, we present the overall scoring function for CIM, formulated as follows:

$$f(\text{vec}(\text{conv}([\mathbf{h}, \mathbf{r}, \mathbf{h}_I, \mathbf{r}_I]))) \mathbf{t}. \quad (11)$$

Overall, the model aims to minimise the same cross-entropy loss for training.

Note that we only take the head entity h as the known entity as an example, and we can also apply our method to a tail entity t when predicting h .

4 Experiments

4.1 Benchmark Dataset

We experiment link prediction task on three commonly accepted benchmarks: FB15k-237, WN18RR and YAGO3-10.

Table 1. Results on link prediction over FB15k-237, WN18RR and YAGO3-10. The best score is in **bold** and second best score is underlined. Since both CIM and FIM generalize CrossE while CIM generalizes ConvTransE alone, we highlight performance comparison among the four methods specially in the table above. Note that Hits@3 is missing for YAGO3-10, as it is not reported by other methods.

| Model | FB15k-237 | | | | | WN18RR | | | | | YAGO3-10 | | | |
|------------|--------------|--------------|--------------|------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| | Hits | | | MR | MRR | Hits | | | MR | MRR | Hits | | MR | MRR |
| | @10 | @3 | @1 | | | @10 | @3 | @1 | | | @10 | @1 | | |
| DistMult | 0.419 | 0.263 | 0.155 | 254 | 0.241 | 0.491 | 0.439 | 0.389 | 5,110 | 0.425 | 0.540 | 0.240 | 5,926 | 0.340 |
| ComplEx | 0.419 | 0.263 | 0.152 | 248 | 0.240 | 0.507 | 0.458 | 0.411 | 5,261 | 0.444 | 0.550 | 0.260 | 6,351 | 0.360 |
| R-GCN | 0.417 | 0.258 | 0.153 | – | 0.248 | – | – | – | – | – | – | – | – | – |
| ConvE | 0.491 | 0.350 | 0.239 | 246 | 0.316 | 0.480 | 0.430 | 0.390 | 5,277 | 0.460 | 0.660 | 0.450 | 2,792 | 0.520 |
| RotatE | 0.533 | 0.375 | 0.241 | 177 | 0.338 | 0.571 | <u>0.492</u> | 0.428 | 3,340 | <u>0.476</u> | – | – | – | – |
| SACN | <u>0.536</u> | <u>0.385</u> | 0.261 | – | 0.352 | 0.540 | 0.480 | 0.430 | – | 0.470 | – | – | – | – |
| InteractE | 0.535 | – | <u>0.264</u> | 172 | <u>0.354</u> | 0.528 | – | 0.430 | 5,202 | 0.463 | 0.687 | 0.462 | <u>2,375</u> | 0.541 |
| CrossE | 0.474 | 0.331 | 0.221 | – | 0.299 | – | – | – | – | – | – | – | – | – |
| ConvTransE | 0.513 | 0.365 | 0.240 | – | 0.331 | 0.520 | 0.470 | 0.430 | – | 0.460 | – | – | – | – |
| FIM (our) | 0.534 | 0.383 | 0.256 | 201 | 0.348 | 0.530 | 0.484 | <u>0.442</u> | <u>4,332</u> | 0.472 | <u>0.688</u> | <u>0.473</u> | 3,017 | <u>0.550</u> |
| CIM (our) | 0.538 | 0.390 | 0.265 | <u>176</u> | 0.355 | <u>0.543</u> | 0.494 | 0.446 | 4,531 | 0.478 | 0.693 | 0.485 | 1,969 | 0.559 |

The FB15k-237 [19] dataset contains extensive knowledge base triples and has been commonly used in link prediction task. Consisting of 14,541 entities and 237 relations, the dataset is a subset of FB15k [3] and all inverse relations are removed.

The WN18RR [4] dataset contains 40,943 entities and 11 relations, derived from WordNet [10] and WN18 [3]. And all inverse relations are removed.

It has been revealed [19] that FB15k and WN18 suffer from test leakage through inverse relations: a large number of test triples can be obtained simply by inverting triples in the training set. To perform the evaluation more rigorously, we prefer FB15k-237 and WN18RR, where inverse relations and inverse triples have been removed.

The YAGO3-10 [9] dataset is a subset of YAGO3 which includes at least 10 relations involved for each entity. YAGO3-10 contains extensive relations with high in-degrees and out-degrees. Since the meanings of these head and tail entities may differ greatly, modeling on YAGO3-10 is more challenging.

4.2 Evaluation Protocol

The reported performance is evaluated on 5 standard metrics: proportion of correct triples ranked at top 1, 3 and 10 (Hits@1, Hits@3, and Hits@10), Mean Rank (MR), and Mean Reciprocal Rank (MRR). The Hits@N and MRR are the higher the better, while the MR is the lower the better. For all experiments, we report averaged results across 5 runs, and we omit the variance on all the metrics as they are not significant.

4.3 Main Results

Performance Comparison. To evaluate our CIM and FIM, we compare them with a range of knowledge graph embedding methods, including current state-of-the-art methods. The results over three benchmark datasets are listed in Table 1.

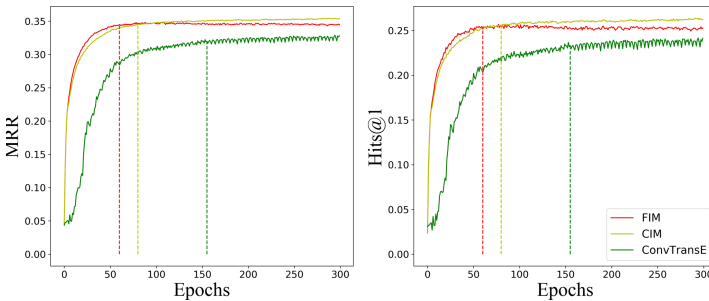


Fig. 4. The convergence study of FIM, CIM and ConvTransE by epochs on FB15k-237 validation set. It takes more than 150 epochs for ConvTransE to reach its convergent performance, but both our proposed CIM and FIM need fewer than 80 epochs. (Color figure online)

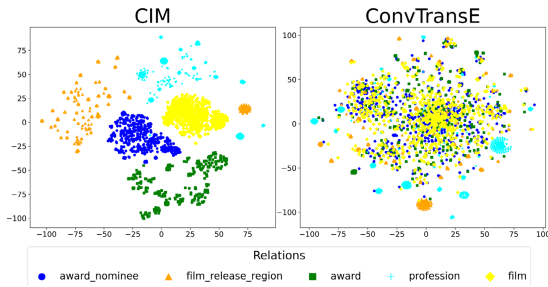
Overall, compared to the existing SOTA models, our CIM outperforms them over all three datasets on Hits@N and MRR, and it also achieves comparable performances on MR. On the other hand, FIM is also competitive with these methods on all metrics over all datasets. This demonstrates the advantages of our proposed methods.

Interaction Mechanism. Since the basic idea of our interaction mechanism is similar to CrossE [24], to explore the interaction between entities and relations, we focus on performance comparisons among FIM, CIM and CrossE. On FB15k-237, both CIM and FIM outperform CrossE on all metrics. On average, our CIM has a 17.5% performance improvement, and our FIM has a 15.2% performance

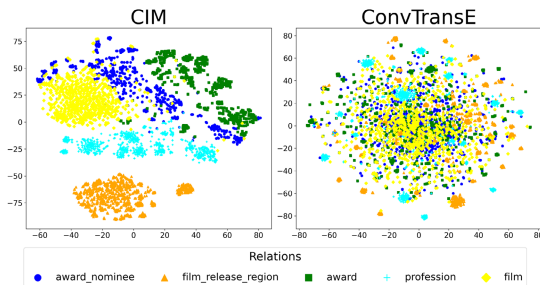
improvement. Based on the improvements of the metrics, we can easily find out our novel interaction mechanism is more effective compared to CrossE, and even our FIM with a simple feed-forward neural network can make a huge boost.

We also compare against ConvTransE [16] with the proposed CIM since CIM also utilises the CNN structures. We observe that CIM outperforms ConvTransE across all metrics. On average, it has a 7.4% performance improvement over FB15k-237 and a 4.2% higher performance over WN18RR. In fact, our CIM adopts the same CNN module as ConvTransE does, and the only difference is that CIM takes our proposed interaction mechanism as the input feature, which aims to strengthen the interactions between entities and relations but cannot be fully learnt by the CNN model, empirically. From the metrics comparison, it is easy to find out our innovative interaction mechanism functions well.

On the other hand, we observe that compared to ConvTransE, our FIM also performs well, which only uses a feed-forward net on our interaction embeddings. On average, it has a 5.2% performance improvement over FB15k-237 and a 2.6% higher performance over WN18RR. This shows that our interaction mechanism can actually encode more information, which cannot be captured by CNN, and that our interaction mechanism is still compatible with CNN models. Even for some metrics, our FIM can outperform the strong CNN baseline ConvTransE, which also demonstrates the superiority of our interacted models.



(a) t-SNE plot of tail entity embeddings learned by CIM (left) and ConvTransE (right).



(b) t-SNE plot of concatenation of head and tail entity embeddings learned by CIM (left) and ConvTransE (right).

Fig. 5. t-SNE plots of learned entity embeddings by our CIM and ConvTransE over FB15k-237 dataset under two settings. We plot 5 most frequent relations with 1,000 entities for each relation.

Comparing with GCN-Based SOTA. As shown in Table 1, SACN [16] is also a competitive baseline and even it is the second best on several metrics. It is built on a weighted graph convolutional network (WGCN) as an encoder, which can help improve performance but at an extra computational expense [16] that it needs to load all the nodes and edges of a graph for training, as well as high computational complexity. By contrast, both of our FIM and CIM are computationally friendly and achieve comparable or even better performance as SACN. On the other hand, our methods are in a fully end-to-end manner, which is much easier for re-training when the KG is modified or evolved. Above all, these observations illustrate the practical and advantages of our models from many perspectives.

4.4 Analysis and Discussion

Convergence Analysis. Figure 4 shows the convergence study of FIM (the red line) and CIM (the yellow line) compared to ConvTransE (the green line) on FB15k-237 validation set. For convergence analysis, we utilize the Adam optimizer [6] and the same settings such as learning rate for all three models. We observe that both our CIM and FIM converge faster than ConvTransE, and achieve better performance. It takes more than 150 epochs for ConvTransE to reach its convergent performance, while both our proposed CIM and FIM need fewer than 80 epochs. The converge difference between our models and ConvTransE proves that our interaction mechanism can provide better representations for models. And after a period of training (after 100 epochs), as is depicted in Fig. 4, our FIM tends to descend slightly, which seems to be overfitting, while the CIM does not. Convincingly, the CNN based CIM is empirically better than the fully-connected layer based FIM.

Can Our Methods Learn Better Embeddings? To understand how well our methods learn interactions between entities and relations, we visualise the learned embeddings. We choose the five most frequent relations and randomly sample 1000 corresponding triples for each relation on FB15k-237 test set. To present in a 2D view, we utilize t-SNE [7] on these learned embeddings. Specially, we compare our CIM with ConvTransE since they both generalise a CNN architecture while CIM has an additional interaction mechanism.

Figure 5a shows the results of using the embeddings of tail entities only. Obviously, grouped by relations, tail entity embeddings trained by our CIM can be easily differentiated and separated from each other, while there is no such pattern for ConvTransE. Since tail entities corresponding to the same relation are supposed to have similar features in the entity space, it is easy to evaluate that the embeddings learned by CIM are better than those learned by ConvTransE in reflecting their semantics.

Figure 5b shows the results of using the concatenated embeddings of both head and tail entities, that is, the concatenation of the triples with the relation masked. Evidently, although the scatters plotted only contain information of entities, our CIM implicates information of relations since clusters of entity

concatenation embeddings corresponding to different relations separate from each other, while ConvTransE can hardly make it. Because the main difference between CIM and ConvTransE lies in our interaction mechanism, we intuitively show the advantages of our interaction mechanism in the embedding process.

5 Conclusion

Introducing a novel interaction mechanism to learn the contextualised representation of entities and relations, we alleviate the limitations of existing KGC methods. Based on the proposed interaction mechanism, we build two effective models, FIM and CIM, utilizing a feed-forward neural network and a convolutional neural network respectively. Through extensive experiments and sufficient comparisons, we demonstrate that considering the contextualised representations of entity and relation, our models achieve the new SOTA performance in link prediction task. Furthermore, we evaluate and prove the necessity of our interaction mechanism in performance improvement.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: DBpedia: a nucleus for a web of open data. In: ISWC 2007 + ASWC 2007 (2007)
2. Bollacker, K.D., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: SIGMOD (2008)
3. 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 (2013)
4. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2D knowledge graph embeddings. In: AAAI (2018)
5. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT (2019)
6. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: Bengio, Y., LeCun, Y. (eds.) ICLR 2015, Conference Track Proceedings (2015)
7. Laurens, V.D.M., Hinton, G.: Visualizing data using t-SNE. *J. Mach. Learn. Res.* **9**(2605) (2008)
8. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: AAAI (2015)
9. Mahdisoltani, F., Biega, J., Suchanek, F.M.: YAGO3: a knowledge base from multilingual Wikipedias. In: CIDR (2015)
10. Miller, G.A.: WordNet: a lexical database for English. *Commun. ACM* **38**(11) (1995)
11. Nathani, D., Chauhan, J., Sharma, C., Kaul, M.: Learning attention-based embeddings for relation prediction in knowledge graphs. In: ACL (2019)
12. Nguyen, T.D., Nguyen, D.Q., Phung, D., et al.: A novel embedding model for knowledge base completion based on convolutional neural network. In: NAACL-HLT (2018)

13. Nickel, M., Tresp, V.: Tensor factorization for multi-relational learning. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) ECML PKDD 2013. LNCS (LNAI), vol. 8190, pp. 617–621. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40994-3_40
14. Nie, Y.P., Han, Y., Huang, J.M., Jiao, B., Li, A.P.: Attention-based encoder-decoder model for answer selection in question answering. *Front. Inf. Technol. Electron. Eng.* **18**(4), 535–544 (2017)
15. Peters, M.E., et al: Deep contextualized word representations. In: NAACL-HLT (2018)
16. Shang, C., Tang, Y., Huang, J., Bi, J., He, X., Zhou, B.: End-to-end structure-aware convolutional networks for knowledge base completion. In: AAAI (2019)
17. Shi, B., Weninger, T.: Open-world knowledge graph completion. In: AAAI (2018)
18. Sun, Z., Vashishth, S., Sanyal, S., Talukdar, P.P., Yang, Y.: A re-evaluation of knowledge graph completion methods. In: ACL (2020)
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 (2015)
20. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: ICML. JMLR Workshop and Conference Proceedings (2016)
21. Vashishth, S., Sanyal, S., Nitin, V., Agrawal, N., Talukdar, P.P.: InteractE: improving convolution-based knowledge graph embeddings by increasing feature interactions. In: AAAI (2020)
22. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: ICLR (2015)
23. Yao, L., Mao, C., Luo, Y.: KG-BERT: BERT for knowledge graph completion. arXiv (2019). <http://arxiv.org/abs/1909.03193>
24. Zhang, W., Paudel, B., Zhang, W., Bernstein, A., Chen, H.: Interaction embeddings for prediction and explanation in knowledge graphs. In: WSDM (2019)
25. Zhao, X., Jia, Y., Li, A., Jiang, R., Song, Y.: Multi-source knowledge fusion: a survey. *World Wide Web* **23**(4), 2567–2592 (2020)