A Re-Ranking Framework for Knowledge Graph Completion

Zikang Wang*†, Linjing Li*‡, Daniel Dajun Zeng*†‡
*The State Key Laboratory of Management and Control for Complex Systems,
Institute of Automation, Chinese Academy of Sciences, Beijing, China
†School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
‡Shenzhen Artificial Intelligence and Data Science Institute (Longhua), Shenzhen, China
{wangzikang2016, linjing.li, dajun.zeng}@ia.ac.cn

***Abstract*—Knowledge graph completion, one of the most important research questions in knowledge graphs, aims at predicting missing links in a given graph. Current mainstream approaches adopt high-quality embeddings of entities and relations of the graph to improve their performances. However, it is not easy to devise a universal embedding learner that can fit various scenarios. In this paper, we propose a general-purpose framework which can be employed to improve the performance of knowledge graph completion. Specifically, given an arbitrary knowledge graph completion model, we first run the original model to get a ranked entity list. Then, we combine the query and the top ranked entities with attention mechanism, re-rank all these entities by feeding the combined vector into a neural network. The proposed re-ranking phase can be conveniently added to a variety of models to improve their performance without substantial modification. We conduct experiments on four datasets: WN18, FB15k, WN18RR, and FB15k-237. We choose TransE, TransH, TransD, DistMult, and ANALOGY as base models. Experiments on these datasets and models validate the effectiveness of the proposed re-ranking framework. We further explore the influence of the number of top ranked entities used in the re-ranking phase. We also test other attention mechanism to determine the most effective one, and found that vanilla attention mechanism can balance accuracy and complexity.**

***Index Terms*—knowledge graph, link prediction, attention mechanism**

I. Introduction

A knowledge graph is a large-scale knowledge base storing relational knowledge between entities in a graph structure. It provides knowledge foundation for a variety of artificial intelligence applications, such as language modeling[1], question answering[2] and machine reading comprehension[3]. However, even the most massive knowledge graph suffers from the problem of incompleteness that many links are missing in the graph, which harms the downstream applications[4]. Knowledge graph completion is an important task that aims to address this problem by predicting missing links in a given knowledge graph.

Most models in the literature try to improve completion performance by proposing new representation learning models to learn more expressive embedding for entities and relations [5], [6], [7], [8], [9], [10]. On the one hand, this leads to models with increasing complexity, such as ConvE [10] and CapsE [11]. On the other hand, different models may have different performance on different datasets, thus it is hard to propose new models to improve performance on a specific dataset in practice.

However, proposing new models may not be the only way to achieve better performance. According to experimental results on existing models, we have noticed that the value of hits@10 metric is quite high. In the knowledge graph completion task, given an entity and the relation in a specific triple (head entity, relation, tail entity), we rank all the entities with their probabilities to be the missing entity. Hits@10 metric indicates the percentage of correct entities among the top 10 ranked entities. Taking TransE [5] and DistMult [8] as examples, Table I illustrates the experimental results, it shows that hits@10 can achieve up to 0.9 on dataset WN18, 0.6 on dataset FB15k, and 0.4 on datasets WN18RR and FB15k-237. Thus, taking advantage of the top ranked entities may enable the model to be aware of more information and bring improvements to existing models.

TABLE I: Hits@10 results of model TransE and DistMult on datasets WN18, FB15k, WN18RR and FB15k-237.

Dataset	Model	Hits@10
WN18	TransE [5]	0.89
	DistMult [8]	0.94
FB15k	TransE [5]	0.47
	DistMult [8]	0.57
WN18RR	TransE [5]	0.41
	DistMult [8]	0.32
FB15k-237	TransE [5]	0.44
	DistMult [8]	0.42

In this paper, we explore the effectiveness of a re-ranking framework, which can be treated as a post-process on the ranking results of existing models. Post-processing is a widely adopted technique in machine learning and data mining, which is simple but can bring improvements to the original models. The proposed framework does not aim to devise any novel embedding learner, otherwise, for a specific model, we first get an initial ranking result based on it, then re-rank all the entities based on the combination of query and top ranked entities with attention mechanism.

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

	Model	Ent. & Rel. Embedding	Score Function
Translation Models	TransE [5]	$v_h, v_t \in \mathbb{R}^d$, $v_r \in \mathbb{R}^d$	$-\|v_h + v_r - v_t\|_{1/2}$
	TransH [6]	$v_h, v_t \in \mathbb{R}^d$, $v_r, w_r \in \mathbb{R}^d$	$-\left\|(v_h - w_r^T v_h w_r) + v_r - (v_t - w_r^T v_t w_r)\right\|_2^2$
	TransR [12]	$v_h, v_t \in \mathbb{R}^d$, $v_r \in \mathbb{R}^k$, $M_r \in \mathbb{R}^{k\times d}$	$-\|M_r v_h + v_r - M_r v_t\|_2^2$
	TransD [7]	$v_h, v_t, w_h, w_t \in \mathbb{R}^d$, $r, w_r \in \mathbb{R}^k$	$-\left\|(w_r w_h^T + I)v_h + v_r - (w_r w_t^T + I)v_t\right\|_2^2$
Bilinear Models	RESCAL [13]	$v_h, v_t \in \mathbb{R}^d$, $M_r \in \mathbb{R}^{d\times d}$	$v_h^T M_r v_t$
	DistMult [8]	$v_h, v_t \in \mathbb{R}^d$, $v_r \in \mathbb{R}^d$	$v_h^T \text{diag}(v_r) v_t$
	ComplEx [14]	$v_h, v_t \in \mathbb{C}^d$, $v_r \in \mathbb{C}^d$	$\text{Re}(v_h^T \text{diag}(r) v_t)$
	ANALOGY [9]	$v_h, v_t \in \mathbb{R}^d$, $M_r \in \mathbb{R}^{d\times d}$	$v_h^T M_r v_t$
Neural Network Models	NTN [15]	$v_h, v_t \in \mathbb{R}^d$, $W_r \in \mathbb{R}^{d\times d\times k}$, $V_r \in \mathbb{R}^{k\times 2d}$, $v_r, b_r \in \mathbb{R}^k$	$v_r^T f(v_h^T W_r v_t + V_r[v_h; v_t] + b_r)$
	ConvE [10]	$v_h, v_t \in \mathbb{R}^d$, $v_r \in \mathbb{R}^d$	$f(vec(f([v_h; v_r] \star \omega))W)v_t$
	ConvKB [16]	$v_h, v_t \in \mathbb{R}^d$, $v_r \in \mathbb{R}^d$	$\text{concat}(f([v_h, v_r, v_t] \star \Omega))w$
	CapsE [11]	$v_h, v_t \in \mathbb{R}^d$, $v_r \in \mathbb{R}^d$	$\|\text{capsnet}(f([v_h; v_r; v_t] \star \Omega))\|_2^2$

TABLE II: Score functions of various models. v_h, v_r, v_t in table are embeddings of head entity h, relation r, and tail entity t respectively, while ω_r, M_r, V_r, b_r are relation-specific parameters.

The proposed re-ranking framework can be easily applied to existing models to further improve their performances, according to our experiments on five existing models through four datasets. We also analyze the influence brought by the different number of top ranked entities we used for re-ranking. We further explore the performance of multi-head attention mechanism but find that it cannot bring much improvements. In pratice, vanilla attention mechanism can balance the accuracy and complexity better.

II. Background

In knowledge graph completion task, an entity is missing in each triple (head entity, relation, tail entity). Given the other entity and the relation, knowledge graph completion aims to predict the missing entity.

Most approaches for knowledge graph completion consist of two phases: embedding learning and ranking. In the embedding learning phase, embeddings in continuous low-dimensional space are learned for entities and relations to embody their latent semantic features. In the ranking phase, each entity provides a candidate triple by replacing the missing entity, scores are calculated for these triples based on the score function, finally, these scores are ranked to generate the most possible missing entity.

The key to both phases in knowledge graph completion is a score function $f : (\mathbb{R}^d, \mathbb{R}^d, \mathbb{R}^d) \to \mathbb{R}$, it assigns each triple a score based on entity and relation embeddings, where d is the dimension of embeddings. After training, scores are learned for triples. In the embedding learning phase, embeddings for entities and relations are learned by maximizing the scores of triples in the knowledge graph, while minimizing scores of triples out of the knowledge graph. In the ranking phase, triples are ranked by scores in descending order. Most approaches improve the performance of completion by proposing new score functions, such as TransE[5], TransH[6], TransR[12], DistMult[8], ANALOGY[9], ConvE[10], and CapsE[11].

These methods can be divided into several categories based on their score functions. Translation models measure plausibility of triples as the distance between two entities with respect to specific relations[17]. Among them TransE[5] is a classical model. It represents all triples (h, r, t) into the same low dimensional space, and measure the distance as $h+r-t$. Since TransE has trouble dealing with complex relations like 1-to-N, N-to-1, and N-to-N, new methods are proposed to overcome this flaw. TransH[6] introduces a relation-specific hyperplane, the distance is calculated based on the projected representation on the hyperplane. TransR[12] introduces specific spaces and calculates distance in the projected space. TransD[7] and TranSparse[18] further simplifies TransR. Other traditional distance models include KG2E[19], UM[20], and SE[21].

Bilinear models measure plausibility of triples based on matching latent semantics of entities and relations by defining score functions as bilinear functions. The first bilinear-based model RESCAL[13] is a semantic matching model, which defines the score of triple (h, r, t) as a bilinear function $f_r(h, t) = h^T M_r t$, where M_r is a relation specific matrix. DistMult[8] proves that a simple bilinear function can achieve outstanding results by restricting the relation specific matrix to be a diagonal matrix. Many bilinear models suffer from the problem that they cannot handle anti-symmetric relations, to solve this problem, ComplEx[14] introduces complex valued embedding to the model. ANALOGY [9] is another model concentrating at the analogical properties of the embedded entities and relations. Other semantic matching models include HolE[22] and so on.

Neural networks are also exploited to perform knowledge graph completion task. NTN[15] uses a neural tensor network to compute scores for triples, ConvE[10] and ConvKB [16] use multi-layer convolutional networks to enhance the expressiveness of the model, CapsE[11] uses a capsule network to model relationship triples to capture variations of the embedding entries at the same dimension of triples. Score functions of some representative models are listed in Table II.

III. Methodology

In this section, we first define the task of knowledge graph completion formally, then describe the re-ranking framework.

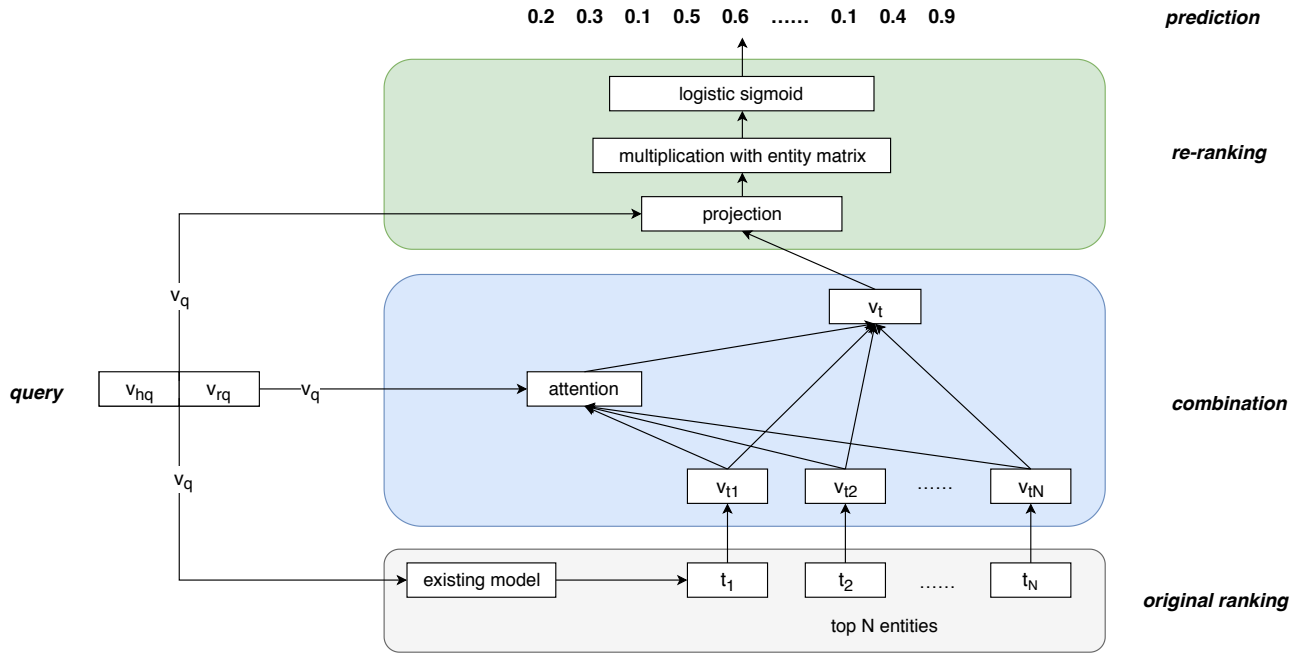

Fig. 1: Architecture of the proposed re-ranking framework.

A. Problem Statement

A knowledge graph is a set of triples, $\mathcal{G} = \{(h, r, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $h, t \in \mathcal{E}$ are entities and $r \in \mathcal{R}$ is a relation, $\mathcal{E}$ and $\mathcal{R}$ are sets of entities and relations. A triple (h, r, t) denotes head entity h and tail entity t have relation r. In this paper, relation r is directional, i.e., the relation r is pointing from h to t.

Knowledge graph completion aims to predict the missing entity in a triple (h, r, t). There are two forms of knowledge graph completion tasks for a given triple (h, r, t), one predicts the tail entity when given head entity h and relation r, denoted as query $(h, r, ?)$, the other predicts the head entity when given tail entity t relation r, denoted as query $(?, r, t)$.

B. Model

The whole framework is shown in Fig. 1 which consists of three steps: original ranking, combination, and re-ranking. To be convenient, we only consider query $(h, r, ?)$ in this section, query $(?, r, t)$ is addressed in the same way.

1) Original Ranking: For an arbitrary knowledge graph completion model $\mathcal{M}$, we first run it to obtain the original ranking and the embeddings of entities and relations. Specifically, for a triple (h, r, t), let (v_h, v_r, v_t) be the learned embeddings respectively. Then a score $f(v_h, v_r, v_t)$ is computed to evaluate the credibility of (h, r, t) based on the learned embedding (v_h, v_r, v_t) and the score function f. The higher the score is, the more trustworthy the triple is.

For a query $q = (h_q, r_q, ?)$, we construct a set $\mathcal{S}'$ of triples by replacing the tail entity by all possible entities $t_i \in \mathcal{E}$,

$$\mathcal{S}' = \{(h_q, r_q, t_i) \mid t_i \in \mathcal{E}\}. \tag{1}$$

For each triple in set $\mathcal{S}'$, a score is then computed based on the score function f and learned embeddings $(v_{h_q}, v_{r_q}, v_{t_i})$. Thus, we can get $|\mathcal{E}|$ scores for the query q, where $|\mathcal{E}|$ is the total number of entities.

In the proposed re-ranking framework, we only take the top ranked entities into consideration. Therefore, we record the top N entities in the ranked list generated by model $\mathcal{M}$,

$$T_q = [t_1, t_2, \ldots, t_N], \ 1 \leq N \leq |\mathcal{E}|, \tag{2}$$

where list T_q is in descending order with t_1 being the entity with the highest score, and N is an integer hyperparameter deciding how many top ranked entities are considered in the re-ranking phase. Intuitively, we merge the information carried by these N entities to further improve the performance of model $\mathcal{M}$.

2) Combination: The combination module, consisting of an attention layer, aggregates information carried by the top N entities with attentions assigned according to the given query q. The overall aggregated vector for top entities is

$$v_t = \sum_{i=1}^{N} \alpha_{qi} v_{t_i}, \tag{3}$$

where v_{t_i} is the embedding of entity $t_i \in \mathcal{E}$, α_{qi} is the attention assigned to entity t_i which is calculated based on the embedding of query v_q and entity v_{t_i} per se. Query embedding v_q is the result of concatenating the embedding of head entity and relation,

$$v_q = v_{h_q} \parallel v_{r_q}, \tag{4}$$

where $\parallel$ is the concatenation operator. Entity embedding v_{t_i} is learned according to model $\mathcal{M}$.

In order to compute α_{qi}, we adopt two settings of the attention mechanism: vanilla attention and multi-head attention.

Vanilla Attention In this setting, attention for an entity, say t_i, is calculated as following,

$$\alpha_{qi} = \frac{\exp\left(g(v_q, v_{t_i})\right)}{\sum_{j=1}^{N} \exp\left(g(v_q, v_{t_j})\right)}, \tag{5}$$

where v_q is the query embedding formed by the embeddings of the given entity h_q and relation r_q, g is a feed-forward neural network.

Multi-head Attention

In this setting, we first update the embeddings of top N entities based on multi-head attention, then combine these embeddings based on vanilla attention and updated embeddings. The multi-head attention layer is used to capture the overall information carried by these entities, it also enables the model to learn different types of information.

The multi-head attention layer first updates embeddings of entities by performing a weighted sum over all N entities. Each v_{t_i} is projected to a key k, value v, and query q with distinct affine transformations following ReLU activation. Denote weight for head H of entity i as $head_{iH}$. The corresponding embedding of key, value, and query of head H of entity i is denoted as k_{iH}, v_{iH}, and q_{iH}, respectively. Then $head_{iH}$ is calculated using the scaled dot-product:

$$head_{iH} = \sum_{j \in [1,K]} v_{jH} \sigma\left(\frac{q_{iH}^T k_{jH}}{\sqrt{d}}\right), \tag{6}$$

where d is the dimension of entities, and $K > 1$ denotes the number of heads, $H \in (1, K]$.

The attention heads $head_{iH}$ are then concatenated and multiplied with weight parameter W^O to generate the updated entity embedding v'_{t_i},

$$v'_{t_i} = \text{Concat}(head_{i1} \parallel \cdots \parallel head_{iK}) W^O. \tag{7}$$

where W^O is a trainable parameter matrix. Then, the attention for each entity can be computed as

$$\alpha_{qi} = \frac{\exp\left(g(v_q, v'_{t_i})\right)}{\sum_{j=1}^{N} \exp\left(g(v_q, v'_{t_j})\right)}. \tag{8}$$

With α_{qi} at hand, the following re-ranking phase is the same for the above two attention settings.

3) Re-ranking: With the help of v_q and v_t computed above, we re-rank T_q to promote the performance of model $\mathcal{M}$. The proposed re-ranker consists of three operations. First, we concatenate v_q and v_t to form a more informative query $(v_q \parallel v_t)$; then, we project $(v_q \parallel v_t)$ into a d-dimensional space by a linear transformation W; finally, we compute new scores and re-rank entities based on $(v_q \parallel v_t)W$. Following are some implementation details.

To accelerate training and evaluation, we take one query and score it against all entities simultaneously as done in [10]. We match new query embeddings with the entity embedding matrix $v_{\mathcal{E}} \in \mathbb{R}^{|\mathcal{E}| \times d}$ via inner product, and choose rectified linear units as score function f. The score vector $\psi(v_q, v_t) \in \mathbb{R}^{1 \times |\mathcal{E}|}$ can be written as

$$\psi(v_q, v_t) = f\left((v_q \parallel v_t)W\right) v_{\mathcal{E}}^T. \tag{9}$$

During the training phase, we apply the logistic sigmoid function $\sigma(\cdot)$ to all scores and minimize the binary cross-entropy loss

$$\mathcal{L}(p, s) = -\frac{1}{N} \sum_i \left(s_i \cdot \log p_i + (1 - s_i) \cdot \log\left(1 - p_i\right)\right), \tag{10}$$

where $p = \sigma\left(\psi_r(v_q, v_t)\right)$ and s is the label vector, $s_i = 1$ if the corresponding triple is true, otherwise $s_i = 0$.

In the evaluation phase, we re-rank all the entities according to the new scores calculated by $\psi(v_q, v_t)$ to form new enhanced ranking result.

IV. Experiments and Results

In this section, we describe the experiments we conducted to evaluate the re-ranking framework and discuss the experimental results.

A. Datasets

We employ four datasets, WN18, FB15k, WN18RR, and FB15k-237, in the following evaluation. Table III lists basic statistics of these datasets.

WN18 and WN18RR are subsets of WordNet [23]. WordNet is a large lexical knowlege graph. It groups words into sets of synsets, each expressing a distinct concept. The synsets are interlinked by means of conceptual-semantic and lexical relations. FB15k and FB15k-237 are subsets of Freebase [24]. Freebase is a large knowlege graph consisting of general facts.

Both WN18 and FB15k are proposed in [5], they are currently the standard benchmarks for knowledge graph completion task. However, they both suffer from the test leakage problem, WN18RR and FB15k-237 are proposed to fix this issue [10]. All four datasets are widely used for evaluation.

TABLE III: Statistics of datasets.

Dataset	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{R}\rvert$	#Train	#Valid	#Test
WN18	40,943	18	141,442	5,000	5,000
FB15k	14,951	1,345	483,142	50,000	59,071
WN18RR	40,943	11	86,835	3,034	3,134
FB15k-237	14,542	237	272,115	17,535	20,466

B. Evaluation Protocol

We follow the same protocol as described in[25]. For tail entity prediction, we first construct all the possible triples by removing the tail entity of each triple and replace it with every entity in the entity set. Triples not in the original knowledge graph is considered as "corrupted" ones, while the original triples are treated as "golden" ones. Then, we rank all the possible triples in descending order according to their scores and record the rank for each golden triple. Head entity prediction is dealt with in the same way.

TABLE IV: Knowledge graph completion results on WN18 and FB15k.

dataset	model	MRR		MR		Hits@10		Hits@3		Hits@1	
		Ori.	Re.	Ori.	Re.	Ori.	Re.	Ori.	Re.	Ori.	Re.
WN18	TransE [5]	.692	**.760**	251	**160**	.891	**.926**	.814	**.902**	.475	**.583**
	TransH [6]	.617	**.673**	388	**310**	.828	**.918**	.765	**.833**	.430	**.442**
	TransD [7]	.701	**.762**	212	**181**	.919	**.934**	.843	**.898**	.517	**.620**
	DistMult [8]	.816	**.894**	890	**735**	**.921**	.910	**.915**	.912	.727	**.819**
	ANALOGY [9]	.938	**.941**	351	**280**	.949	**.958**	.942	**.951**	.935	**.940**
FB15k	TransE [5]	.417	**.561**	125	**64**	.476	**.701**	.314	**.523**	.150	**.413**
	TransH [6]	.495	**.525**	**87**	90	.641	**.745**	.535	**.640**	.284	**.347**
	TransD [7]	.523	**.579**	91	**84**	.777	**.829**	.624	**.670**	.389	**.423**
	DistMult [8]	.649	**.660**	**97**	101	.831	**.840**	.729	**.798**	.550	**.641**
	ANALOGY [9]	.729	**.780**	81	**71**	.847	**.859**	.766	**.772**	.623	**.650**

TABLE V: Knowledge graph completion results on WN18RR and FB15k-237.

dataset	model	MRR		MR		Hits@10		Hits@3		Hits@1	
		Ori.	Re.	Ori.	Re.	Ori.	Re.	Ori.	Re.	Ori.	Re.
WN18RR	TransE [5]	.181	**.185**	**5107**	5670	**.410**	.390	**.326**	.239	.024	**.093**
	TransH [6]	.144	**.145**	5871	**5230**	.356	**.383**	.274	**.281**	.003	**.058**
	TransD [7]	.139	**.141**	**5699**	6283	.362	**.374**	.259	**.281**	.004	**.052**
	DistMult [8]	.178	**.184**	7801	**7389**	**.324**	.268	**.201**	.195	.102	**.111**
	ANALOGY [9]	.120	**.134**	**7998**	8423	.218	**.236**	.129	**.178**	.072	**.128**
FB15k-237	TransE [5]	.272	**.283**	348	**302**	.443	**.449**	.300	**.313**	.186	**.197**
	TransH [6]	.136	**.254**	**374**	387	.331	**.426**	.160	**.288**	.041	**.179**
	TransD [7]	.174	**.251**	**387**	397	.358	**.414**	.202	**.270**	.083	**.169**
	DistMult [8]	.268	**.280**	285	**276**	.421	**.431**	.293	**.305**	.191	**.211**
	ANALOGY [9]	.229	**.274**	423	**332**	.377	**.427**	.220	**.310**	.136	**.199**

With the ranks of golden triples, three metrics are adopted to evaluate the performance: mean rank (MR), mean reciprocal rank (MRR), and Hits@N. MR is the average of predicted ranks for golden triples; MRR is the average of inverse ranks for golden triples [10]; Hits@N is the proportion of golden entities ranked in top N. We use Hits@1, Hits@3, and Hits@10 [10]. For these metrics, lower MR, higher MRR, and higher Hits@N are preferred.

C. Experimental Setup

We evaluate the proposed framework on five models: TransE [5], TransH [6], TransD [7], DistMult [8], and ANALOGY [9]. TransE, TransH and TransD are translation models, DistMult and ANALOGY are bilinear models. We aim to enhance performance of these simple models, so we do not perform experiments on neural network models.

We train these models to get the original rank using OpenKE[1] [26] with parameters reported in the original papers. Though the final results are not exactly the same as those reported in the papers, it is still fair since we do not make comparisons among these models, we compare the performance before and after re-ranking instead.

[1]https://github.com/thunlp/OpenKE

In the experiments, we use the same neural network architecture for both tail-prediction query $(h, r, ?)$ and head-prediction query $(?, r, t)$, but train them separately. The dimension of entities and relations is set as 100, the optimizer is Adam, the learning rate is set to be 0.01, and the times of training is bounded by 40. In the re-ranking phase, the top 10 entities from the original ranking are adopted. We use vanilla attention mechanism to balance the accuracy and complexity, this will be further discussed in the next Section.

D. Experimental Results

Table IV reports both the original results (Ori.) and the re-ranked results (Re.) on datasets WN18 and FB15k. Table V reports the results on datasets WN18RR and FB15k-237. These two tables show that, in most cases, the proposed re-ranking framework can be employed to further improve the original results. We notice that, the improvement on the hits@1 metric is especially stable. This is an advantage if only one answer entity is desired in some applications. However, the improvements on both Hits@N and MRR may worsen the MR metric, and we leave this question for future work.

TABLE VI: Knowledge graph completion results with different number of heads in multi-head attention.

# heads		Original	0	2	4	6	8	10
model TransH on dataset WN18RR								
Metrices	MRR	.144	.145	.142	.146	.147	.149	.148
	MR	5871	5230	5223	5210	5204	5202	5209
	Hits@10	.356	.383	.383	.385	.388	.392	.386
	Hits@3	.274	.281	.280	.282	.285	.289	.284
	Hits@1	.003	.058	.057	.059	.061	.065	.060
model TransH on dataset FB15k-237								
Metrices	MRR	.136	.254	.256	.260	.262	.270	.268
	MR	374	387	379	382	378	372	375
	Hits@10	.331	.426	.428	.429	.432	.435	.432
	Hits@3	.160	.288	.290	.289	.291	.293	.293
	Hits@1	.041	.179	.182	.183	.185	.186	.182

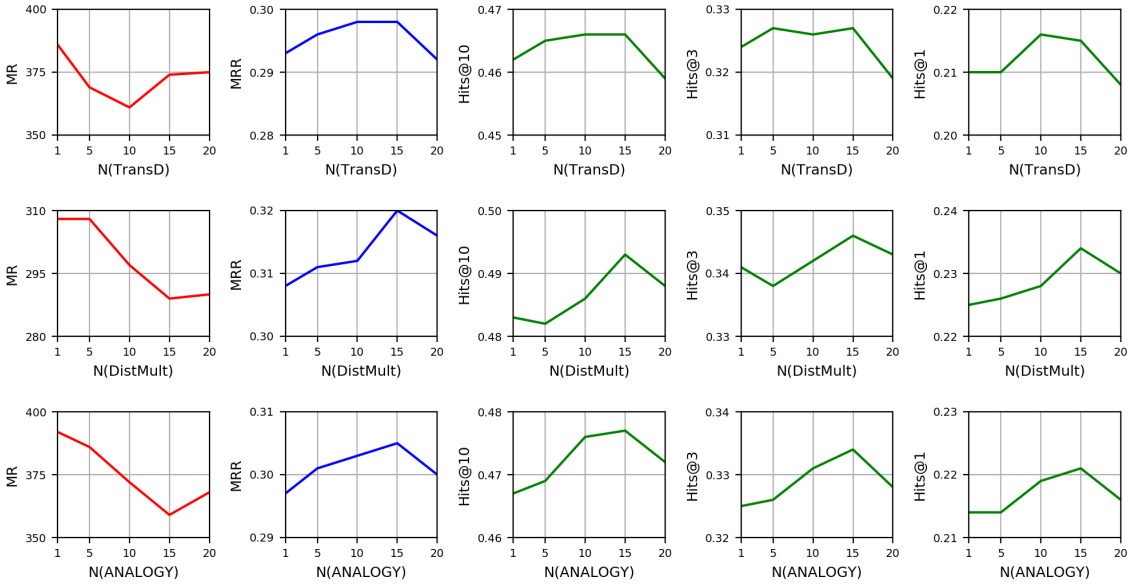

Fig. 2: Knowledge graph completion results of TransD, DistMult and ANALOGY(in 3 rows) when N equals $1, 5, 10, 15, 20$ (in 5 rows) respectively.

V. Analysis

In this section, we further analyze the influence of different attention mechanisms and the number of top entities we used in the re-ranking phase.

A. Influence of Attention Mechanism

We further explore whether more complicated attention mechanism can bring larger improvements in the proposed framework. To do so, we apply multi-head attention and test different number of attention heads. We evaluate the performance of TransH on datasets WN18RR and FB15k-237 when number of heads equals $0, 2, 4, 6, 10$ respectively. Results under different attention mechanisms are shown in Table VI. It shows that number of heads has little influence on performance. Considering the high complexity of multi-head attention, we use vanilla attention to balance accuracy and complexity in our model.

B. Influence of the Number of Top Entities

A key hyperparameter in the re-ranking framework is N, the number of top ranked entities used in the re-ranking phase. It may affect both the performance and complexity of the model. To explore its influence on the model performance, we conduct comparative experiments on three baselines: TransD, DistMult, and ANALOGY. For each of these models, we train tail entity

prediction models on dataset FB25k-237, the parameter N is controlled as 1, 5, 10, 15, and 20. All other parameters are kept the same. Fig. 2 illustrates the results. Results of one model are illustrated in a line, values for the same metric are illustrated in a column.

Increasing N from 1 to 20, the performance first improves and then deteriorates. This is consistent with the intuition that too few top entities may not carry sufficient knowledge, while too many may bring irrelevant noise to the model.

Last but not least, the optimal value of N for different models may be different. In our experiments, the optimum is 10 for TransD and 15 for both DistMult and ANALOGY.

VI. Conclusions and Future Work

In this paper, we proposed a re-ranking framework to improve the performance of knowledge graph completion of base methods based on re-ranking with attention mechanism. In this framework, we first ranked entities using the original models, then combined the top ranked entities with query entity and relation through attention mechanism, and re-ranked entities based on the combined vector. The proposed framework is flexible that can be applied to various models without substantive modification. We validated the proposed framework on four datasets and five existing base models. In the experiments, the proposed framework achieved better results than the original counterparts. We further analyzed the influence of different attention mechanisms and the number of top entities used in the re-ranking phase.

Our framework, though can be easily applied to most models, still has limitations. When applying this framework on some models, such as ComplEx [14], which learns embeddings in a complex space, i.e., v_h, v_r, and v_t are vectors in space $\mathbb{C}^d$, the structure of the neural network needs to be modified. We will focus on this question in our future work and improve its generalization performance.

Acknowledgment

This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFC0820105 and 2016QY02D0305, the National Natural Science Foundation of China under Grants 71702181, 71621002, as well as the Key Research Program of the Chinese Academy of Sciences under Grant ZDRW-XH-2017-3. Linjing Li is the corresponding author.

References

[1] R. Logan, N. F. Liu, M. E. Peters, M. Gardner, and S. Singh, "Barack's wife hillary: Using knowledge graphs for fact-aware language modeling," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 5962–5971. [Online]. Available: https://www.aclweb.org/anthology/P19-1598

[2] W. Xiong, M. Yu, S. Chang, X. Guo, and W. Y. Wang, "Improving question answering over incomplete KBs with knowledge-aware reader," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 4258–4264. [Online]. Available: https://www.aclweb.org/anthology/P19-1417

[3] A. Yang, Q. Wang, J. Liu, K. Liu, Y. Lyu, H. Wu, Q. She, and S. Li, "Enhancing pre-trained language representations with rich knowledge for machine reading comprehension," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 2346–2357. [Online]. Available: https://www.aclweb.org/anthology/P19-1226

[4] M. Schlichtkrull, T. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings*. Greece: Springer, 2018, pp. 593–607.

[5] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in Neural Information Processing Systems 26*. Lake Tahoe, USA: Curran Associates, Inc., 2013, pp. 2787–2795.

[6] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, ser. AAAI'14. Canada: AAAI Press, 2014, pp. 1112–1119.

[7] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Beijing, China: Association for Computational Linguistics, Jul. 2015, pp. 687–696.

[8] B. Yang, W. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, USA, 2015.

[9] H. Liu, Y. Wu, and Y. Yang, "Analogical inference for multi-relational embeddings," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70. International Convention Centre, Sydney, Australia: PMLR, 06–11 Aug 2017, pp. 2168–2178.

[10] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. USA: AAAI Press, 2018, pp. 1811–1818.

[11] D. Q. Nguyen, T. Vu, T. D. Nguyen, D. Q. Nguyen, and D. Phung, "A Capsule Network-based Embedding Model for Knowledge Graph Completion and Search Personalization," in *Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019, pp. 2180–2189.

[12] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, ser. AAAI'15. Austin, Texas: AAAI Press, 2015, pp. 2181–2187.

[13] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ser. ICML'11. Bellevue, Washington, USA: Omnipress, 2011, pp. 809–816. [Online]. Available: http://dl.acm.org/citation.cfm?id=3104482.3104584

[14] T. Trouillon, J. Welbl, S. Riedel, Éric Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," in *Proceedings of The*

33rd International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, vol. 48. New York, USA: PMLR, 20–22 Jun 2016, pp. 2071–2080.

[15] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Lake Tahoe, USA: Curran Associates, Inc., 2013, pp. 926–934. [Online]. Available: http://papers.nips.cc/paper/5028-reasoning-with-neural-tensor-networks-for-knowledge-base-completion.pdf

[16] D. Q. Nguyen, T. D. Nguyen, D. Q. Nguyen, and D. Phung, "A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network," in *Proceedings of the 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2018, pp. 327–333.

[17] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications." *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 12, pp. 2724–2743, 2017.

[18] G. Ji, K. Liu, S. He, and J. Zhao, "Knowledge graph completion with adaptive sparse transfer matrix," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16. AAAI Press, 2016, pp. 985–991. [Online]. Available: http://dl.acm.org/citation.cfm?id=3015812.3015959

[19] S. He, K. Liu, G. Ji, and J. Zhao, "Learning to represent knowledge graphs with gaussian embedding," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, ser. CIKM '15. New York, NY, USA: ACM, 2015, pp. 623–632. [Online]. Available: http://doi.acm.org/10.1145/2806416.2806502

[20] A. Bordes, X. Glorot, J. Weston, and Y. Bengio, "Joint learning of words and meaning representations for open-text semantic parsing," in *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, N. D. Lawrence and M. Girolami, Eds., vol. 22. La Palma, Canary Islands: PMLR, 21–23 Apr 2012, pp. 127–135. [Online]. Available: http://proceedings.mlr.press/v22/bordes12.html

[21] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, "Learning structured embeddings of knowledge bases," in *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, ser. AAAI'11. AAAI Press, 2011, pp. 301–306. [Online]. Available: http://dl.acm.org/citation.cfm?id=2900423.2900470

[22] M. Nickel, L. Rosasco, and T. Poggio, "Holographic embeddings of knowledge graphs," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16. Phoenix, Arizona: AAAI Press, 2016, pp. 1955–1961. [Online]. Available: http://dl.acm.org/citation.cfm?id=3016100.3016172

[23] G. A. Miller, "Wordnet: a lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[24] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '08. New York, NY, USA: ACM, 2008, pp. 1247–1250.

[25] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, "Learning structured embeddings of knowledge bases," in *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, ser. AAAI'11. San Francisco, California: AAAI Press, 2011, pp. 301–306.

[26] X. Han, S. Cao, L. Xin, Y. Lin, Z. Liu, M. Sun, and J. Li, "Openke: An open toolkit for knowledge embedding," in *Proceedings of EMNLP*, 2018.