2018 24th International Conference on Pattern Recognition (ICPR)
Beijing, China, August 20-24, 2018

A Novel Asymmetric Embedding Model for Knowledge Graph Completion

Zhiqiang Geng, Zhongkun Li,Yongming Han*

College of Information Science and Technology, Beijing University of Chemical Technology
Engineering Research Center of Intelligent PSE, Ministry of Education in China
Beijing, China
E-mail address: hanym@mail.buct.edu.cn

***Abstract*—Modeling knowledge graph completion by encoding each entity and relation into a continuous tensor space becomes very hot. Meanwhile, many models including TransE, TransH, TransR, CTransR, TransD, TranSpare, TransDR, STransE, DT, FT and OrbitE are proposed for knowledge graph completion. However, all these previous works take less attention to the asymmetrical and the imbalance of many relations (some relations link a subject and many objects, and other relations link many subjects and many objects). Therefore, this paper proposes a novel asymmetrical embedding model(AEM) for knowledge graph completion. Because of the different properties of the head and tail entities in the triplets of the same relationship, every head entity vector and every tail entity vector are weighted by the corresponding head relation vector and the corresponding tail relation vector, respectively. And then new entity vector representations are obtained and the new entity vectors in the same triple are similar. Because the AEM weights each dimension of the entity vectors, it can accurately represent the latent attributes of entities and relationships. Moreover, the number of parameters of the AEM is so small that it is easier to train. Finally, compared with previous embedding models, the AEM obtains a better link prediction performance through two benchmark datasets FB15K and WN18.**

***Keywords*—knowledge graph; representation learning; knowledge graph completion; Asymmetrical Embedding**

I. INTRODUCTION

Knowledge graph is a semantic network consisting of a large number of entities and their rich relations. Meanwhile, The knowledge graph plays an important role in many artificial intelligence applications, such as relation extraction(RE)[1], question answering(Q&A)[2], knowledge inference[3], information retrieval[4], etc. A general knowledge graph is described as the multi-relational data and stored in form of triplets with the head entity, the tail entity and the relationship between the head entity and the tail entity. In recent years, with the development of the big data, various large-scale knowledge graphs such as WordNet[5], Freebase[6], Yago[7] and NELL[8] have been built either collaboratively or (partly) automatically. However, due to the large-scale data and the rapid increase of knowledge graphs, these knowledge graphs often suffer from incompleteness. Knowledge graph completion aims to predict missing entities and relations among entities based on existing triplets in an incomplete knowledge graph. However, it is almost impossible to predict all relations(including some latent relations) among entities by using the traditional model based on logic and symbol[9][10]. Therefore, a novel embedding model is proposed to embed the entities and relations among entities into a continuous, real-valued and low-dimensional vector space through representation learning models. Compared with classical models, the feasibility and robustness of the embedding model have been verified.

Among various embedding models, such as Semantic Matching Energy(SME)[25], Neural Tensor Network(NTN)[26], RESCAL[27], translation-based models can achieve state-of-the-art prediction performance. The most typical translation-based model was TransE[11], which learned the low-dimensional embedding for every entity and relation in knowledge graphs. And for a gold triplet (*h, r, t*), the embedding $\boldsymbol{h}$ of the head entity h is closed to the embedding $\boldsymbol{t}$ of the tail entity t by adding the embedding $\boldsymbol{r}$ of relation r, which is $\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{t}$. The TransE model is simple and effective on the benchmark tasks of the link prediction and the triple classification in general cases. However, it is not good at dealing with complex relations which are reflexive, transitive, 1-to-many, many-to-1, or many-to-many. In order to solve these problems, various extended models are proposed. TransH[12] regards a relation as a translating operation on a relation-specific hyperplane, which is characterized by a norm vector $\boldsymbol{w_r}$ and translation vector $\boldsymbol{d_r}$. The embedding $\boldsymbol{h}$ and the embedding $\boldsymbol{t}$ are projected to the hyperplane of relation r to obtain vectors $\boldsymbol{h_r}= \boldsymbol{h} - \boldsymbol{w_r^T h w_r}$ and $\boldsymbol{t_r} = \boldsymbol{t} - \boldsymbol{w_r^T t w_r}$, and then $\boldsymbol{h_r} + \boldsymbol{d_r} \approx \boldsymbol{t_r}$. Since TransH made each entity to have distinct representations in different relations, the performance on complex relations can be achieved better than the TransE. However, entities and relations are different objects, so it is insufficient to model them in the same space. The TransR/CTransR model[13] first projected entity vectors from entity space into the corresponding relation space through the matrix $\boldsymbol{M_r}$, and then built translating operations between projected entities when $\boldsymbol{M_r h} + \boldsymbol{r} \approx \boldsymbol{M_r t}$. However, the types and attributes of entities linked by the same relation are various, so this model cannot let the head entity and the tail entity share the same mapping space. Meanwhile, there are too many parameters in the TransR/CTransR model. Moreover, the TransD model[14] used two distinct projection matrices to respectively project the head entity and the tail entity into the relation space. Meanwhile, The TranSparse model[15] replaced the dense matrices of the TransR model based on sparse matrices. And the STransE[16] projected head entity vectors and tail entity vectors from entity space into two relation space through matrixes $\boldsymbol{M}_{rh}$ and $\boldsymbol{M}_{rt}$ (the essence of this baseline is to combine the TransR model and the SE[21] model), and then built translating operations between projected entities when $\boldsymbol{M}_{rh}\boldsymbol{h} + \boldsymbol{r} \approx \boldsymbol{M}_{rt}\boldsymbol{t}$. Furthermore, DT[17], FT[18], TransDR[19] and OrbitE[20] were also proposed.

In this paper, a novel asymmetrical embedding model(AEM) for knowledge graph completion is proposed. And the primary contribution of the AEM is that in a triple, each dimension of the entity vectors is weighted by each dimension of the relation vectors, and then the similarity of the weighted head entity vector and the weighted tail entity vector is compared. Meanwhile, every relation can be regarded as a vector. However, the head entity and the tail entity in a triple have various types and attributes, so every relation should be presented as two vectors $\boldsymbol{r}_h$, $\boldsymbol{r}_t$. And then the vector of the head entity $\boldsymbol{h}$ is weighted by $\boldsymbol{r}_h$ and the vector of the tail entity $\boldsymbol{t}$ is weighted by $\boldsymbol{r}_t$. Finally, the proposed knowledge graph completion model is $<\boldsymbol{h}, \boldsymbol{r}_h> \approx <\boldsymbol{t}, \boldsymbol{r}_t>$ ($<\boldsymbol{e}, \boldsymbol{r}>$ stands for multiplication by elements of the two vectors). The AEM weights the head entity and the tail entity based on different relation sub-vectors, so the AEM has a better effect on one-to-many, many-to-one, many-to-many relationships. Moreover, because there are smaller number of parameters in AEM than other basic models, it is easy to train and not prone to overfit.

Finally, the AEM performs better than other state-of-the-art link prediction models through two standard link prediction datasets WN18 and FB15K, so it can serve as a new baseline for the knowledge graph completion.

II. PROPOSED METHOD

Let $\boldsymbol{E}$ and $\boldsymbol{T}$ denote the entities set and the relations set, respectively. For each triple (h, r, t), where $h, t \in \boldsymbol{E}$ and $r \in \boldsymbol{T}$. The score function of the AEM is defined in Eq. (1).

$$f_r(h, t) = \|<\boldsymbol{h}, \boldsymbol{r}_h> - <\boldsymbol{t}, \boldsymbol{r}_t>\|_{l1/2} \qquad (1)$$

$$<\boldsymbol{e}, \boldsymbol{r}> = \textbf{vector}(e_1 r_1,\ e_2 r_2,\ e_3 r_3, \ldots\ldots,\ e_k r_k)$$

$$\boldsymbol{e}, \boldsymbol{r}, \boldsymbol{r}_h, \boldsymbol{r}_t, \boldsymbol{h}, \boldsymbol{t} \in R^k$$

For the score function (1), the l_1 norm or the l_2 norm is used. In the experiments, it is proved that the l_1 norm gives a slightly better result.

To learn the vectors of entities and relations, the margin-based objective function is shown in Eq.(2).

$$L = \sum_{(h, r, t) \in S} \sum_{(h, r, t) \in S'} [\gamma + f_r(h, t) - f_r(h', t')]_+ \qquad (2)$$

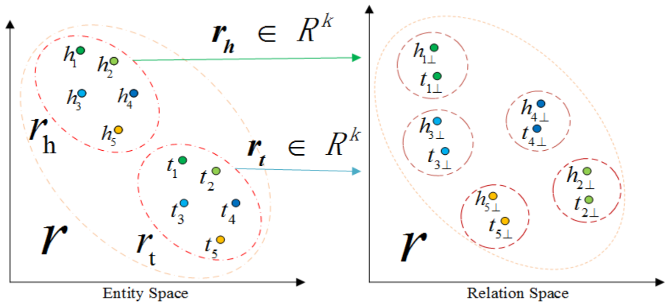
$$S' = \{(h', r, t) | h' \in \boldsymbol{E}, (h', r, t) \notin S\} \cup \{(h, r, t') | t' \in \boldsymbol{E}, (h, r, t') \notin S\} \qquad (3)$$

Where $[x]_+ = \max(0, x)$, γ is the margin hyperparameter, S is the training set consisting of correct triplets. And Eq.(3) is the set of incorrect triples generated by corrupting a correct triplet $(h, r, t) \in S$. The loss function (2) is minimized by the way of stochastic gradient descent(SGD), and imposed the following constraints during training:

$$\|\boldsymbol{h}\| \le 1,\ \|\boldsymbol{t}\| \le 1,\ \|\boldsymbol{r}_h\| \le 1,\ \|\boldsymbol{r}_t\| \le 1.$$

It is worth noting that the function $[x]_+$ itself is non-differentiable. However, if $x<0$, then $[x]_+ = 0$, and the operation of the SGD is skipped. If $x>0$, then $[x]_+ = x$, and the operation of the SGD can be proceeded to update the trained parameters of the AEM.

The AEM is simply shown Fig 1.



$\boldsymbol{r}_h \in R^k$
$\boldsymbol{r}_t \in R^k$
h_1 h_2 h_3 h_4 h_5
t_1 t_2 t_3 t_4 t_5
r_h
r_t
r
$h_{1\perp}$ $t_{1\perp}$ $h_{3\perp}$ $t_{3\perp}$ $h_{4\perp}$ $t_{4\perp}$ $h_{2\perp}$ $t_{2\perp}$ $h_{5\perp}$ $t_{5\perp}$
r
Entity Space
Relation Space

Fig 1. Simple illustration of the AEM.

In Fig 1, the r stands for an entity set in the current relational space. r_h stands for the head entity set in the current relational space and r_t stands for the tail entity set in current relational space. The head entity set and the tail entity set are weighted by the relation vector $\boldsymbol{r}_h \in R^k$ and $\boldsymbol{r}_t \in R^k$, respectively. And then the new head entities and the new tail entities, the vectors of which are similar in the same triple, are obtained.

The score function of the AEM is shown in Eq. (1)., which is different from the score functions of the TransE model, the TransR model and the STransE model shown in Eq.(4)-(6), respectively.

The score function of the TransE model is:

$$f_r(h, t) = \|\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{t}\|_{l1/2} \qquad (4).$$

The score function of the TransR model is

$$f_r(h, t) = \|\boldsymbol{M}_r\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{M}_r\boldsymbol{t}\|_{l1/2} \qquad (5).$$

The score function of the STransE model is

$$f_r(h, t) = \|\boldsymbol{M}_{rh}\boldsymbol{h} + \boldsymbol{r} - \boldsymbol{M}_{rt}\boldsymbol{t}\|_{l1/2} \qquad (6).$$

The score function of the AEM replaces the matrixes that project the entities from entity space to relation space with the relation vectors, So the AEM is simple and easy to train.

The detailed optimization procedure is described in as follows.

Algorithm about the AEM

Require: Training set $S = \{(h, r, t)\}$, entity set $\boldsymbol{E}$ and relation set $\boldsymbol{T}$, margin γ, embedding dim k, the SGD learning rate τ.

Process:

Entity embeddings $\{\boldsymbol{e}\}$ and relation embedding $\{\boldsymbol{r}_h, \boldsymbol{r}_t\}$ for head and tail entities.

```
1: Initialize e ← uniform($-\frac{6}{\sqrt{k}}$, $\frac{6}{\sqrt{k}}$) for each entity $e \in E$
2:          $r_h, r_t$ ← uniform($-\frac{6}{\sqrt{k}}$, $\frac{6}{\sqrt{k}}$) for each entity $r \in T$
3:          $r_h, r_t \leftarrow r_h/\|r_h\|,\ r_t/\|r_t\|$ for each entity $r \in T$
4: end initialize
5: Loop
6:     $e \leftarrow e/\|e\|$ for each entity $e \in E$
```

```
7:  $S_{batch} \leftarrow sample(S, b)$ //sample a minibatch of size b
8:  $Tr_{batch} \leftarrow \emptyset$ // initialize the set of pairs of triplets
9:  for $(h, r, t) \in S_{batch}$ do
10:   $(h', r, t') \leftarrow sample(S'_{(h, r, t)})$//sample a corrupted triplet
11:   $Tr_{batch} \leftarrow Tr_{batch} \cup \{((h, r, t),(h', r, t'))\}$
12:  end for
13:  Update embeddings w.r.t.
14:   $\sum_{((h, r, t),(h', r, t')) \in Tr_{batch}} \nabla[\gamma + f_r(h, r) - f_r(h', t')]_+$
15:  for $s \in$ entities or relations in $Tr_{batch}$ do
16:   if $||s||>1$ then
17:     $s \leftarrow s/||s||$//constrains $||e||\le 1$, $||r_h||\le 1$, $||r_t||\le 1$
18:   end if
19:  end for
20: end loop
```

Notes: The bold and slanted lowercases (i.e., $\boldsymbol{r_h}$, $\boldsymbol{r_t}$) stand for vectors (or named as embedding), the bold and slanted uppercases (i.e., $\boldsymbol{E}$, $\boldsymbol{T}$) stand for sets.

In the AEM algorithm, all embeddings for entities and relationships are first initialized following the random procedure proposed in[11]. Meanwhile, the embedding vectors of the entities in each iteration of the AEM are first normalized. Then a subset of triplets is sampled from the training set to serve as the training set of this iteration. For each triplet, a single corrupted triplet is sampled. The parameters are updated by the SGD of each iteration. Finally, the AEM is stopped based on a validation set.

III. COMPARISON OF COMPLEXITY ABOUT SEVERAL MODELS

The complexities of some basic models and the AEM are listed in TABLE I.

TABLE I. Comparison of time complexity and space complexity of different models

Model	Space complexity	Time complexity
SE[21]	$O(N_e m+2N_r n^2)$	$O(2m^2 N_t)$
TransE[11]	$O(N_e m+N_r n)$	$O(N_t)$
TransH[12]	$O(N_e m+2N_r n)$	$O(2mN_t)$
TransR[13]	$O(N_e m+N_r (m+1)n)$	$O(2mnN_t)$
TransD[14]	$O(2N_e m+2N_r n)$	$O(2nN_t)$
TranSpare[15]	$O(2N_e m+2N_r (m+1)(1-\vartheta)n)(0\ll\vartheta\le 1)$	$O(2(1-\vartheta)mnN_t)(0\ll\vartheta\le 1)$
TranSpare-DT[17]	$O(2N_e m+2N_r (m+1)(1-\vartheta)n)(0\ll\vartheta\le 1)$	$O((2n-2\vartheta n+3)mN_t)(0\ll\vartheta\le 1)$
STransE[16]	$O(2N_e m+N_r (2m+1)n)$	$O(2mnN_t)$
TransDR[19]	$O(N_e m+3N_r n)$	$O(3nN_t)$
OrbitE Sphere[20]	$O(N_e m+N_r (n+1))$	$O((m+1)N_t)$
OrbitE Hilbert[20]	$O(N_e m+N_r (n+1))$	$O((9m+1)N_t)$
OrbitE Hyperplane[20]	$O(N_e m+N_r (2n+1))$	$O((3m+1)N_t)$
AEM(this paper)	$O(N_e m+2N_r n)$	$O(2mN_t)$

Notes: N_e and N_r represent the number of entities and relations, respectively. N_t represents the number of triples in a knowledge graph. m is the dimension of entity embedding space and n is the dimension of relationship embedding space.

The time complexity of the AEM is the same as the transH model and lower than some other baselines such as SE, TransR, TranSpare, TranSpare-DT, STransE,TransDR, OrbitE Hilbert and OrbitE Hyperplane. The space complexity of the AEM is lower than some other baselines such as TransD, TranSpare, TranSpare-DT, STransE, TransDR and OrbitE Hilbert.

IV. EXPRIMENTS AND RESULTS ANALYSIS

For the link prediction evaluation, some experiments are conducted and the performance of the AEM will be compared with published results on the benchmark WN18 and FB15K datasets that are extracted from real knowledge graphs of WordNet and Freebase, respectively. The information about these datasets is given in TABLE II.

TABLE II. Statistics of the datasets

Dataset	#Ent	#Rel	#Train	#Valid	#Test
WN18	40,943	18	141,442	5,000	5,000
FB15K	14,951	1,345	483,142	50,000	59,071

Notes: #Ent is the number of entities, #Rel is the number of relation types, and #Train, #Valid and #Test are the number of the training, validation and test sets, respectively.

Task and evaluation protocol

The link prediction task aims to predict the missing entities. An alternative of the entities and a relation are given while the embedding models infer the other missing entity, i.e. predicting *h* given (*?, r, t*) or predicting *t* given (*h, r, ?*) where *?* denotes the missing entity. The results are evaluated using the ranking induced by the score function $f_r(h, t)$ on test triplets

Evaluation Protocol. Each test triple (*h, r, t*) is corrupted by replacing either *h* or *t* with each of the possible entities. Then an implausibility score of this corrupted triple with the score function $f_r(h, t)$ is calculated. After ranking these scores in descending order, the rank of the original triple is obtained. Following the protocol described in the previous studies[11], the evaluation metric is the proportion of the testing triplet whose rank is not larger than T(Hits@T) for each model. Hits@10 is used for common completion and reasoning ability and Hits@1 is used for more accurate reasoning, which is called "Raw". After the corrupted triples that exist in the training set, validation set, or test set are filtered out, the Hits@T is calculated again, which is called "Filter". However, in order to exclude the fact that a corrupted triple is in the knowledge graph and it is ranked ahead the original triple, "Filter" is the more preferred indicator. In both "Raw" and "Filter", the higher Hits@T, the better the performance.

Implementation. the experimental results of other baselines are reproduced from the literature for Hits@10 and Hits@1. However, when some results of these baselines are not present in the literature, they will be automatically ignored. The "Bernoulli" trick is applied for generating head or tail entities when sampling incorrect triples. The optimal configurations of the AEM are as followed through repeated experiments. On WN18, for Hits@10 and Hits@1, the number of vector dimensions k=400, the margin hyper-parameter $\gamma = 0.5$, and the SGD learning rate τ=0.01. On FB15K, for Hits@10, the number of vector dimensions k=1000, the margin hyper-parameter $\gamma = 0.25$, and the SGD learning rate $\tau = 0.01$. Meanwhile, for Hits@1, the number of vector dimensions k=1000, the margin hyper-parameter $\gamma = 0.20$, and the SGD learning rate $\tau = 0.01$. The experimental environment is a PowerEdge T630 server with E5-2640 CPU of 2.60 GHz, 32G Memory and Operating System of Ubuntu 16.04 Server.

Main Experimental Results

The link prediction results of the AEM and several baselines are shown in TABLE III.

TABLE III. Evaluation results on link prediction

Datasets	**WN18**			**FB15K**		
Metric	*Hits@10(%)*		*Hits@1(%)*	*Hits@10(%)*		*Hits@1(%)*
	Raw	*Filter*	*Filter*	*Raw*	*Filter*	*Filter*
SE[21]	68.5	80.5	--	28.8	39.8	--
TransE[11]	75.4	89.2	29.5	34.9	47.1	24.4
TransH[12]	75.4	86.7	31.3	45.7	64.4	24.8
TransR[13]	79.8	92.0	5.2	48.2	68.7	20.0
TransD[14]	79.9	92.6	--	53.4	77.3	--
TranSparse[15]	80.5	93.9	--	53.7	79.9	--
TranSpare-DT[17]	80.4	94.3	--	53.9	80.2	--
TransDR[19]	79.9	93.2	--	55.3	78.2	--
STransE[16]	--	93.4	--	--	79.7	--
OrbitE Sphere[20]	80.7	92.8	55.8	55.2	81.6	48.9
OrbitE Hilbert[20]	79.2	90.2	40.5	**55.7**	86.7	57.0
OrbitE Hyperplane[20]	**84.2**	93.2	90.6	55.2	**87.4**	69.6
AEM(this paper)	82.0	**95.1**	**93.3**	53.7	86.5	**69.7**

On WN18, the AEM is better than other baselines on the indicators of Hits@10 for "Filter" setting and Hits@1 for "Filter" setting. And on the indicator of Hits@10 for "Raw" setting, the AEM is second only to the model of OrbitE Hyperplane. On FB15K, the AEM is better than other baselines on the indicators of Hits@1 for "Filter" setting. On the indicator of Hits@10 for "Filter" setting, the AEM is lower than the model of OrbitE Sphere and OrbitE Hyperplane, but still higher than other baselines. On the indicators of Hits@10 for "Raw" setting, the AEM does not perform very well and is 2% lower than the model of OrbitE Hilbert, but it is higher than the model of SE, TransE, TransH, TransR and TransD. On the indicator of the Hits@1 for "Filter" setting, the AEM is higher than other baselines on both WN18 and FB15K datasets.

The analysis results of Hits@10 and Hits@1 on FB15K with respect to the relation categories defined are shown in TABLE IV and TABLE V, respectively.

TABLE IV. Evaluation results on FB15K by mapping properties of relations (%)

Tasks	**Predicting Head (Hits@10(%))**	**Predicting Tail (Hits@10(%))**

Relation Category	*1-to-1*	*1-to-many*	*many-to-1*	*many-to-many*	*1-to-1*	*1-to-many*	*many-to-1*	*many-to-many*
SE[21]	35.6	62.6	17.2	37.5	34.9	14.6	68.3	41.3
TransE[11]	43.7	65.7	18.2	47.2	43.7	19.7	66.7	50.0
TransH[12]	66.8	87.6	30.2	64.5	65.5	39.8	83.3	67.2
TransR[13]	78.8	89.2	34.1	69.2	79.2	38.4	90.4	72.1
TransD[14]	86.1	95.5	47.1	78.5	85.4	50.6	94.4	81.5
TranSparse[15]	87.5	95.9	51.8	81.2	87.6	60.0	94.5	83.7
TranSpare-DT[17]	87.4	95.8	51.9	81.6	86.7	59.9	94.8	84.0
TransDR[19]	86.3	94.9	44.1	80.3	86.2	52.7	94.0	81.7
STransE[16]	82.8	94.2	50.4	80.1	82.4	56.9	93.4	83.1
OrbitE Sphere[20]	86.3	**97.0**	43.1	78.8	84.7	56.5	**95.8**	81.8
OrbitE Hilbert[20]	**92.4**	95.9	49.2	84.4	81.6	64.7	94.2	88.6
OrbitE Hyperplane[20]	91.4	96.9	54.9	86.2	**93.4**	65.5	95.4	89.8
AEM(this paper)	91.3	96.0	**56.0**	**88.4**	91.9	**66.8**	94.7	**90.5**

TABLE V. Evaluation precise results on FB15K by mapping properties of relations (%)

Tasks	**Prediction Head (Hits@1(%))**				**Prediction Tail (Hits@1(%))**			
Relation Category	*1-to-1*	*1-to-many*	*many-to-1*	*many-to-many*	*1-to-1*	*1-to-many*	*many-to-1*	*many-to-many*
TransE[11]	35.4	50.7	8.6	18.1	34.5	10.6	56.1	20.3
TransH[12]	35.3	48.7	8.4	16.9	35.5	10.4	57.5	19.3
TransR[13]	29.5	42.8	6.1	14.5	28.0	7.7	44.1	16.2
OrbitE Sphere[20]	35.7	72.1	18.1	34.6	41.4	30.0	75.5	41.3
OrbitE Hilbert[20]	52.7	70.8	30.9	52.9	50.8	32.6	72.2	57.3
OrbitE Hyperplane[20]	65.5	88.4	38.6	66.0	63.7	49.2	82.4	69.1
AEM(this paper)	**75.6**	**88.9**	**42.8**	**69.8**	**75.0**	**57.1**	**82.9**	**71.4**

For each kind of relation *r*, the average number N_{rh} of heads *h* for a pair (*r, t*) and the average number N_{rt} of tails *t* for a pair (*h, t*) are calculated. If $N_{rh}<1.5$ and $N_{rt}<1.5$, the relation *r* is labeled as 1-to-1. If $N_{rh}<1.5$ and $N_{rt}\geq1.5$, the relation *r* is labeled as 1-to-many. If $N_{rh}\geq1.5$ and $N_{rt}<1.5$, the relation *r* is labeled as many-to-1. If $N_{rh}\geq1.5$ and $N_{rt}\geq1.5$, the relation *r* is labeled as many-to-many.

The predicted results on the indicator of Hits@10 in different types of relations are described in TABLE IV. The prediction accuracy of the AEM is better than other baselines when predicting head entities of many-to-1 and many-to-many relation category. Meanwhile. the prediction accuracy of the AEM is better than other baselines when predicting tail entities of 1-to-many and many-to-many relation category. For the rest of the metrics, the accuracy of predicting head entities of the 1-to-1 relation category and the 1-to-many relation category based on the AEM is third only to OrbitE Hilbert and OrbitE Hyperplane and only to OrbitE Sphere and OrbitE Hyperplane, respectively. Meanwhile, the accuracy of predicting head entities of the 1-to-1 relation category and is second only to OrbitE Hyperplane. And the accuracy of predicting head entities of the many-to-1 relation category based on the AEM is fourth only to OrbitE Sphere, OrbitE Hyperplane and TranSpare-DT.

The predicted results on the indicator of Hits@1 in different types of relations are described in TABLE V. It can be seen from TABLE V that the predicted results of the AEM are better than other baselines on all the indictors. Especially, when predicting head entities of 1-to-1 relation category, the AEM gets 10.1% improvement over the model of OrbitE Hyperplane. And when predicting tail entities of 1-to-1 relation category, the AEM gets 11.3% improvement over the model of OrbitE Hyperplane. Meanwhile, when predicting tail entities of 1-to-many relation category, the AEM gets 7.9% improvement over the model of OrbitE Hyperplane. And when predicting head entities of many-to-1 relation category, the AEM gets 4.2% improvement over the model of OrbitE Hyperplane. Moreover, when predicting head entities of many-to-many relation category, the AEM gets 3.8% improvement over the model of OrbitE Hyperplane. Furthermore, it can be seen from TABLE III that the AEM is superior to other basic models on the indictors for Hits@1. Therefore, through these experimental results, it is proved that the AEM can be more detailed and concise to depict the intrinsic attributes of complexed entities and diverse relations.

V. CONCUSION AND FUTURE WORK

This paper proposes the AEM to capture more semantic information on complex entities and diverse relations. For a triple (*h, r, t*), the AEM weights the vectors of the head entity *h* and the tail entity *t* based on the sub-vector ($\boldsymbol{r_h}$, $\boldsymbol{r_t}$) of the corresponding relationship *r*, respectively. Meanwhile, the AEM can accurately determine the intrinsic properties of entities and relations in view of the different properties of the head entities and the tail entities. Moreover, the number of parameters and the time complexity of the AEM are lower than other baselines. The experimental results show that the AEM achieves the significant improvement in the link prediction subtask. Furthermore, the AEM outperforms other basic models on the subtask of predicting entities for Hits@1 in the different relations.

The AEM does not describe entities and relations through some additional keywords and does not consider the hierarchy of relations. In the future, the DKRL model[22] and the RTransE model[23] can be imitated to the asymmetrical embedding model. Meanwhile, a new TorusE model for knowledge graph completion (accepted by AAAI-18)[24] is proposed, so we will apply this good idea of the TorusE model into the AEM. Moreover, the AEM will be considered to insert into a framework or model for relation extraction from text.

ACKNOWLEDGMENT

This research was partly funded by National Natural Science Foundation of China (61673046, 61603025), the Natural Science Foundation of Beijing, China (4162045) and the National Key Research and Development Program of China (2017YFC1601800).

REFERENCES

[1] R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld, Knowledge-based weak supervision for information extraction of overlapping relations. in Proc. ACL, Portland, OR, USA, 2011, pp. 541–550.

[2] A. Bordes, J. Weston, and N. Usunier, Open question answering with weakly supervised embedding models. in Proc. ECML PKDD, Nancy, France, 2014, pp. 165–180.

[3] M. Fan, Q. Zhou, and T. F. Zheng. Learning Embedding Representations for Knowledge Inference on Imperfect and Incomplete Repositories. Ieee/wic/acm International Conference on Web Intelligence IEEE, 2017,pp.42-48.

[4] F. Corcoglioniti, M. Dragoni, M. Rospocher, and A. Palmero Aprosio, Knowledge extraction for information retrieval. in Proc. 13th Int. Conf. Semantic Web, 2016, pp. 317–333.

[5] G. A. Miller, WordNet: A lexical database for English, Commun. ACM, vol. 38, no. 11, Nov. 1995,pp. 39–41.

[6] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, Freebase: A collaboratively created graph database for structuring human knowledge. in Proc. SIGMOD, Vancouver, BC, Canada, 2008, pp. 1247–1250.

[7] F. M. Suchanek, G. Kasneci, and G. Weikum, Yago: A core of semantic knowledge. in Proc. WWW, Banff, AB, Canada, 2007, pp. 697–706.

[8] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, Jr., and T. M. Mitchell, Toward an architecture for never-ending language learning. in Proc. AAAI, Atlanta, GA, USA, 2010, pp. 1306–1313.

[9] N. Lao, T. Mitchell, and W. W. Cohen, Random walk inference and learning in a large scale knowledge base. in Proc. EMNLP, Edinburgh,U.K., 2011, pp. 529–539.

[10] W. Y. Wang, K. Mazaitis, N. Lao, and W. W. Cohen, Efficient inference and learning in a large knowledge base. Mach. Learn., vol. 100, no. 1, Apr. 2015,pp. 101–126.

[11] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko,''Translating embeddings for modeling multi-relational data,'' in Proc.NIPS, Lake Tahoe, NV, USA, 2013, pp. 2787–2795.

[12] Z. Wang, J. Zhang, J. Feng, and Z. Chen, Knowledge graph embedding by translating on hyperplanes. in Proc. AAAI, Quebec City, QC, Canada,2014, pp. 1112–1119.

[13] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, Learning entity and relation embeddings for knowledge graph completion. in Proc. AAAI, Austin, TX,USA, 2015, pp. 2181–2187.

[14] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, Knowledge graph embedding via dynamic mapping matrix. in Proc. ACL, Beijing, China, 2015,pp. 687–696.

[15] G. Ji, K. Liu, S. He, and J. Zhao, Knowledge graph completion with adaptive sparse transfer matrix. in Proc. AAAI, Phienix, AZ, USA, 2016, pp. 985–991.

[16] Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. STransE: a novel embedding model of entities and relationships in knowledge bases. in Proc. NAACL.2016, pp. 460–466

[17] L. Chang, M. Zhu, T. Gu, C. Bin, J. Qian, and J. Zhang, Knowledge graph embedding by dynamic translation. IEEE Access,2017, 5, 20898-20907.

[18] J. Feng, M. Huang, M. Wang, M. Zhou, Y. Hao, and X. Zhu, ''Knowledge graph embedding by flexible translation,'' in *Proc. KR*, Cape Town,South Africa, 2016, pp. 557–560.

[19] Z. Tan, X. Zhao, Y. Fang, W. Xiao, and J. Tang,. Knowledge Representation Learning via Dynamic Relation Spaces. IEEE, International Conference on Data Mining Workshops, 2017,pp.684-691.

[20] H. Xiao, M.L. Huang, and X.Y. Zhu, From one point to a manifold: Knowledge graph embedding for precise link prediction. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016, pp. 1315–1321.

[21] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, ''Learning structured embeddings of knowledge bases,'' in *Proc. AAAI*, San Francisco, CA,USA, 2011, pp. 301–306.

[22] R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun. Representation Learning of Knowledge Graphs with Entity Descriptions .in *Proc. AAAI*, 2016, pp. 189–205.

[23] A. Garc´ıa-Duran, A. Bordes, and N. Usunier. Composing Relationships with Translations. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, pp. 286–290.

[24] T. Ebisu, and R. Ichise, TorusE: Knowledge Graph Embedding on a Lie Group.2017, arXiv preprint arXiv:1711.05435. accepted by AAAI-18, https://arxiv.org/abs/1711.05435.

[25] A. Bordes, X. Glorot, J. Weston and Y. Bengio, Joint learning of words and meaning representations for open-text semantic parsing. In International Conference on Artificial Intelligence and Statistics,pp. 127–135.

[26] R. Socher, D. Chen, C. D. Manning, and A. Y. Ng, Reasoning With Neural Tensor Networks for Knowledge Base Completion. In Proc. NIPS, pp. 926-934.

[27] M. Nickel, V. Tresp, H.P. Kriegel, A three-way model for collective learning on multi-relational data. International Conference on International Conference on Machine Learning. Omnipress, 2011,pp. 809-816.