



# MSGE: A Multi-step Gated Model for Knowledge Graph Completion

Chunyang Tan<sup>(✉)</sup>, Kaijia Yang, Xinyu Dai, Shujian Huang, and Jiajun Chen

National Key Laboratory for Novel Software Technology,  
Nanjing University, Nanjing, China

{tancy,yangkj}@smail.nju.edu.cn, {daixinyu,huangs,j,chenjj}@nju.edu.cn

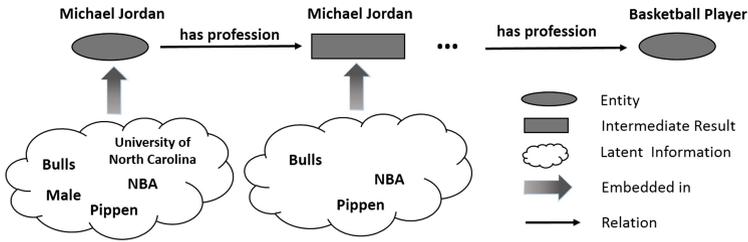
**Abstract.** Knowledge graph embedding models aim to represent entities and relations in continuous low-dimensional vector space, benefiting many research areas such as knowledge graph completion and web searching. However, previous works do not consider controlling information flow, which makes them hard to obtain useful latent information and limits model performance. Specifically, as human beings, predictions are usually made in multiple steps with every step filtering out irrelevant information and targeting at helpful information. In this paper, we first integrate iterative mechanism into knowledge graph embedding and propose a multi-step gated model which utilizes relations as queries to extract useful information from coarse to fine in multiple steps. First gate mechanism is adopted to control information flow by the interaction between entity and relation with multiple steps. Then we repeat the gate cell for several times to refine the information incrementally. Our model achieves state-of-the-art performance on most benchmark datasets compared to strong baselines. Further analyses demonstrate the effectiveness of our model and its scalability on large knowledge graphs.

**Keywords:** Knowledge graph embedding · Gate mechanism · Multi-step

## 1 Introduction

Large-scale knowledge graphs(KGs), such as Freebase [1], YAGO3 [2] and DBpedia [3], have attracted extensive interests with progress in artificial intelligence. Real-world facts are stored in KGs with the form of (subject entity, relation, object entity), denoted as  $(s, r, o)$ , benefiting many applications and research areas such as question answering and semantic searching. Meanwhile, KGs are still far from complete with missing a lot of valid triplets. As a consequence, many researches have been devoted to knowledge graph completion task which aims to predict missing links in knowledge graphs.

Knowledge graph embedding models try to represent entities and relations in low-dimensional continuous vector space. Benefiting from these embedding models, we can do complicated computations on KG facts and better tackle the



**Fig. 1.** Example of link prediction in knowledge graph. An entity has much latent information entailed in its embedding, but with a given relation, only part of them are helpful for predicting.

KG completion task. Translation distance based models [4–8] regard predicting a relation between two entities as a translation from subject entity to tail entity with the relation as a media. While plenty of bilinear models [9–13] propose different energy functions representing the score of its validity rather than measure the distance between entities. Apart from these shallow models, deeper models [14, 15] are proposed to extract information at deep level.

Though effective, these models do not consider: **1.** Controlling information flow specifically, which means keeping relevant information and filtering out useless ones, as a result restricting the performance of models. **2.** The multi-step reasoning nature of a prediction process. An entity in a knowledge graph contains rich latent information in its representation. As illustrated in Fig. 1, the entity *Michael Jordon* has much latent information embedded in the knowledge graph and will be learned into the representation implicitly. However, when given a relation, not all latent semantics are helpful for the prediction of object entity. Intuitively, it is more reasonable to design a module that can capture useful latent information and filter out useless ones. At the meantime, for a complex graph, an entity may contain much latent information entailed in an entity, one-step predicting is not enough for complicated predictions, while almost all previous models ignore this nature. Multi-step architecture [16, 17] allows the model to refine the information from coarse to fine in multiple steps and has been proved to benefit a lot for the feature extraction procedure.

In this paper, we propose a **Multi-Step Gated Embedding (MSGE)** model for link prediction in KGs. During every step, gate mechanism is applied several times, which is used to decide what features are retained and what are excluded at the dimension level, corresponding to the multi-step reasoning procedure. For partial dataset, gate cells are repeated for several times iteratively for more fine-grained information. All parameters are shared among the repeating cells, which allows our model to target the right features in multi-steps with high parameter efficiency. We do link prediction experiments on 6 public available benchmark datasets and achieve better performance compared to strong baselines on most datasets. We further analyse the influence of gate mechanism and the length of steps to demonstrate our motivation.

## 2 Background

### 2.1 Link Prediction in Knowledge Graphs

Link prediction in knowledge graphs aims to predict correct object entities given a pair of subject entity and relation. In a knowledge graph, there are a huge amount of entities and relations, which inspires previous work to transform the prediction task as a scoring and ranking task. Given a known pair of subject entity and relation  $(s, r)$ , a model needs to design a scoring function for a triple  $(s, r, o)$ , where  $o$  belongs to all entities in a knowledge graph. Then model ranks all these triples in order to find the position of the valid one. The goal of a model is to rank all valid triples before the false ones.

### 2.2 Knowledge Graph Embedding

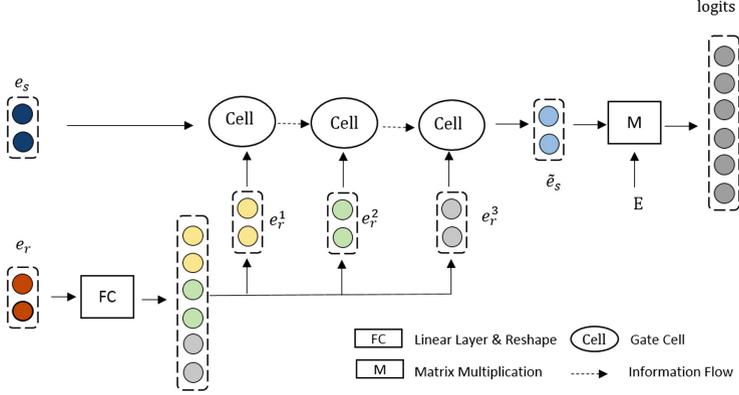
Knowledge graph embedding models aim to represent entities and relations in knowledge graphs with low-dimensional vectors  $(\mathbf{e}_s, \mathbf{e}_r, \mathbf{e}_t)$ . TransE [4] is a typical distance-based model with constraint formula  $\mathbf{e}_s + \mathbf{e}_r - \mathbf{e}_t \approx 0$ . Many other models extend TransE by projecting subject and object entities into relation-specific vector space, such as TransH [5], TransR [6] and TransD [18]. TorusE [7] and RotatE [8] are also extensions of distance-based models. Instead of measuring distance among entities, bilinear models such as RESCAL [9], DistMult [10] and ComplEx [11] are proposed with multiplication operations to score a triplet. Tensor decomposition methods such as Simple [12], CP-N3 [19] and TuckER [13] can also be seen as bilinear models with extra constraints. Apart from above shallow models, several deeper non-linear models have been proposed to further capture more underlying features. For example, (R-GCNs) [15] applies a specific convolution operator to model locality information in accordance to the topology of knowledge graphs. ConvE [14] first applies 2-D convolution into knowledge graph embedding and achieves competitive performance.

The main idea of our model is to control information flow in a multi-step way. To our best knowledge, the most related work to ours is TransAt [20] which also mentioned the two-step reasoning nature of link prediction. However, in TransAt, the first step is categorizing entities with Kmeans and then it adopts a distance-based scoring function to measure the validity. This architecture is not an end-to-end structure which is not flexible. Besides, error propagation will happen due to the usage of Kmeans algorithm.

## 3 Methods

### 3.1 Notations

We denote a knowledge graph as  $\mathcal{G} = \{(s, r, o)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ , where  $\mathcal{E}$  and  $\mathcal{R}$  are the sets of entities, relations respectively. The number of entities in  $\mathcal{G}$  is  $n_e$ , the number of relations in  $\mathcal{G}$  is  $n_r$  and we allocate the same dimension  $d$  to entities and relations for simplicity.  $\mathbf{E} \in \mathbb{R}^{n_e * d}$  is the embedding matrix for



**Fig. 2.** The schematic diagram of our model with length of step 3.  $e_s$  and  $e_r$  represent embedding of subject entity and relation respectively.  $e_r^i$  means the query relation are fed into the  $i$ -th step to refine information.  $\tilde{e}_s$  is the final output information, then matrix multiplication is operated between  $\tilde{e}_s$  and embedding matrix of entities  $E$ . At last, logistic sigmoid function is applied to restrict the final score between 0 and 1.

entities and  $R \in \mathbb{R}^{n_r * d}$  is the embedding matrix for relations.  $e_s$ ,  $e_r$  and  $e_o$  are used to represent the embedding of subject entity, relation and subject entity respectively. Besides, we denote a gate cell in our model as  $C$ .

### 3.2 Multi-step Gate Mechanism

In order to obtain useful information, we need a specific module to extract needed information from subject entity with respect to the given relation, which can be regarded as a control of information flow guided by the relation. To model this process, we introduce gate mechanism, which is widely used in data mining and natural language processing models to guide the transmission of information, e.g. Long Short-Term Memory (LSTM) [21] and Gated Recurrent Unit (GRU) [22]. Here we adopt gating mechanism at dimension level to control information entailed in the embedding. To make the entity interact with relation specifically, we rewrite the gate cell in multi-steps with two gates as below:

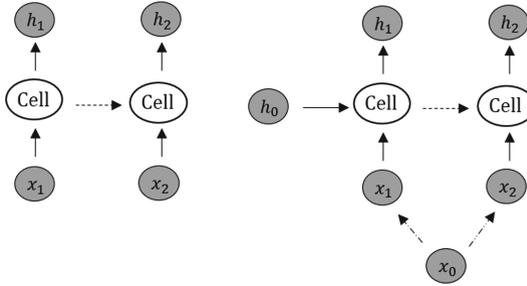
$$\begin{aligned} z &= \sigma(\mathbf{W}_z[e_r, e_s] + \mathbf{b}_z) \\ r &= \sigma(\mathbf{W}_r[e_r, e_s] + \mathbf{b}_r) \end{aligned} \quad (1)$$

Two gates  $z$  and  $r$  are called update gate and reset gate respectively for controlling the information flow. Reset gate is designed for generating a new  $e'_s$  or new information in another saying as follows:

$$e'_s = \tanh(\mathbf{W}_s[r \odot e_s, e_r] + \mathbf{b}) \quad (2)$$

Update gate aims to decide how much the generated information are kept according to formula (3):

$$\tilde{e}_s = (1 - z) \odot e'_s + z \odot e_s \quad (3)$$



**Fig. 3.** The differences between traditional RNN-like model and our model. In RNN-like model (left),  $h_0$  is initialized randomly,  $x$  represents a sequence. In our model(right),  $h_0$  comes from subject entity and  $x$  is transformed from relation  $x_0$ .

Hardmard product is performed to control the information at a dimension level. The values of these two gates are generated by the interaction between subject entity and relation.  $\sigma$ -Logistic sigmoid function is performed to project results between 0 and 1. Here 0 means totally excluded while 1 means totally kept, which is the core module to control the flow of information. We denote the gate cell as  $C$ .

Besides, to verify the effectiveness of gate mechanism, we also list the formula of a cell that exclude gates as below for ablation study:

$$\tilde{e}_s = \tanh(\mathbf{W}[e_r, e_s] + \mathbf{b}) \quad (4)$$

With the gate cell containing several gating operations, the overall architecture in one gate cell is indeed a multi-step information controlling way.

### 3.3 Iterative Multi-step Architecture

In fact, a single gate cell can generate useful information since the two gating operations already hold great power for information controlling. However, for a complex dataset, more fine and precise features are needed for prediction. The iterative multi-step architecture allows the model to refine the representations incrementally. During each step, a query is fed into the model to interact with given features from previous step to obtain relevant information for next step. As illustrated in Fig. 2, to generate the sequence as the input for multi-step training, we first feed relation embedding into a fully connected layer:

$$e'_r = \mathbf{W}e_r + \mathbf{b} \quad (5)$$

We reshape the output as a sequence  $[e_r^0, e_r^1, \dots, e_r^k] = \text{Reshape}(e'_r)$  which are named query relations. This projection aims to obtain query relations of different latent aspects such that we can utilize them to extract diverse information across multiple steps. Information of diversity can increase the robustness of a

model, which further benefits the performance. Query relations are fed sequentially into the gate cell to interact with subject entity and generate information from coarse to fine. Parameters are shared across all steps so multi-step training are performed in an iterative way indeed.

Our score function for a given triple can be summarized as:

$$\phi = (C^k(e_s, [e_r^0, e_r^1, \dots, e_r^k]))e_o \quad (6)$$

where  $C^k$  means repeating gate cell for  $k$  steps and during each step only the corresponding  $e_r^i$  is fed to interact with output information from last step. See Fig. 2 for better understanding. After we extract the final information, it is interacted with object entity with a dot product operation to produce final score.

**Differences with RNN-like Model.** In previous RNN-like models, a cell is repeated several times to produce information of an input sequence, where the repeating times are decided by the length of the input sequence. Differently, we have two inputs  $e_s$  and  $e_r$  with totally different properties, which are embeddings of subject entity and relation respectively, which should not be seen as a sequence as usual. As a result, a gate cell is used for capturing interactive information among entities and relations iteratively in our model, rather than extracting information of just one input sequence. See Fig. 3 for differences more clearly.

**Training.** At last, matrix multiplication is applied between the final output information and embedding matrix  $E$ , which can be called 1-N scoring [14] to score all triples in one time for efficiency and better performance. We also add reciprocal triple for every instance in the dataset which means for a given  $(s, r, t)$ , we add a reverse triple  $(t, r^{-1}, s)$  as the previous work. We use binary cross-entropy loss as our loss function:

$$\mathcal{L}(p, y) = -\frac{1}{N} \sum_i (y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)) \quad (7)$$

We add batch normalization to regularise our model and dropout is also used after layers. For optimization, we use Adam for a stable and fast training process. Embedding matrices are initialized with xavier normalization. Label smoothing [23] is also used to lessen overfitting.

## 4 Experiments

In this section we first introduce the benchmark datasets used in this paper, then we report the empirical results to demonstrate the effectiveness of our model. Analyses and ablation study are further reported to strengthen our motivation.

**Table 1.** Statistics of datasets.

Dataset	#Entity	#Rel	#Train	#Valid	#Test
WN18	40943	18	141442	5000	5000
WN18RR	40943	11	86835	3034	3134
FB15k	14951	1345	483142	50000	59071
FB15k-237	14541	237	212115	17535	20466
UMLS	135	49	5216	652	661
KINSHIP	104	26	8544	1068	1074

#### 4.1 Datasets

Six datasets are used in our experiments:

- **WN18** [4] is extracted from WordNet describing the hierarchical structure of words, consisting relations such as *hyponym* and *hypernym*.
- **WN18RR** [4, 14] is a subset of WN18 which removes inverse relations. Inverse relation pairs are relations such as (*hyponym*, *hypernym*). Inverse relations may cause severe test leakage: a lot of test triples can be obtained from train data simply by inverting them. That means a simple rule-based model can easily figure out the right  $o$  given a  $(s, r)$ , only if it has seen  $(o, r', s)$  in the train data and it knows  $r'$  is the reverse of  $r$ .
- **FB15k** [4] is extracted from Freebase describing mostly relations about movies, actors, awards, sports and so on.
- **FB15k-237** [24] is a subset of FB15k which removes inverse relations and the triples involved in train, valid and test data.
- **UMLS** [25] comes from biomedicine. Entities in UMLS (Unified Medical Language System) are biomedical concepts such as *disease* and *antibiotic*.
- **Kinship** [25] contains kinship relationships among members of the Alyawarra tribe from Central Australia.

The details of these datasets are reported in Table 1.

#### 4.2 Experiment Setup

The evaluation metric we use in our paper includes Mean Reciprocal Rank(MRR) and Hit@K. MRR represents the reciprocal rank of the right triple, the higher the better of the model. Hit@K reflects the proportion of gold triples ranked in the top K. Here we select K among  $\{1, 3, 10\}$ , consistent with previous work. When Hit@K is higher, the model can be considered as better. All results are reported with 'Filter' setting which removes all gold triples that have existed in train, valid and test data during ranking. We report the test results according to the best performance of MRR on validation data as the same with previous works.

**Table 2.** Link prediction results on WN18, WN18RR, FB15k and FB15k-237. † denotes the results are taken from [14]; - denotes the results are not provided; The results of RotatE [8] are reported without self-adversarial negative sampling, this sampling trick is irrelevant with the model itself; Other results are all taken from the original paper. Best results are in bold. The second best are underlined.

Model	WN18		WN18RR		FB15k		FB15k-237	
	MRR	Hit@10	MRR	Hit@10	MRR	Hit@10	MRR	Hit@10
TransE (2013)	-	0.892	-	-	-	0.471	-	-
DistMult† (2015)	0.822	0.936	0.430	0.490	0.654	0.824	0.241	0.419
ComplEx† (2016)	0.941	0.947	0.440	0.510	0.692	0.840	0.247	0.428
R-GCN (2017)	0.814	<b>0.964</b>	-	-	0.696	0.842	0.248	0.417
TransAt (2018)	-	0.951	-	-	-	0.782	-	-
MINERVA (2018)	-	-	0.448	0.513	-	-	0.293	0.456
ConvE (2018)	0.943	0.956	0.430	0.520	0.657	0.831	0.325	0.501
TorusE (2018)	0.947	0.954	-	-	0.733	0.832	-	-
RotatE (2019)	-	-	-	-	-	-	0.297	0.480
Simple (2018)	0.942	0.947	-	-	0.727	0.838	-	-
TuckER (2019)	<b>0.953</b>	0.958	<b>0.470</b>	<u>0.526</u>	<u>0.795</u>	<u>0.892</u>	<b>0.358</b>	<u>0.544</u>
MSGE(Ours)	<u>0.951</u>	<u>0.961</u>	<u>0.464</u>	<b>0.547</b>	<b>0.806</b>	<b>0.894</b>	<u>0.357</u>	<b>0.545</b>

**Table 3.** Link prediction results on UMLS and Kinship.

Model	UMLS				Kinship			
	MRR	Hit@10	Hit@3	Hit@1	MRR	Hit@10	Hit@3	Hit@1
ComplEx <sup>∇</sup> (2016)	0.894	<b>0.995</b>	0.962	0.824	0.838	0.980	0.910	0.754
ConvE <sup>∇</sup> (2018)	0.933	0.992	0.964	0.894	0.797	0.974	0.886	0.697
NTP (2017)	0.872	0.970	0.906	0.817	0.612	0.777	0.700	0.500
NeuralLP (2017)	0.778	0.962	0.869	0.643	0.619	0.912	0.707	0.475
MINERVA (2018)	0.825	0.968	0.900	0.728	0.720	0.924	0.812	0.605
MSGE(Ours)	<b>0.946</b>	0.993	<b>0.973</b>	<b>0.914</b>	<b>0.865</b>	<b>0.988</b>	<b>0.941</b>	<b>0.785</b>

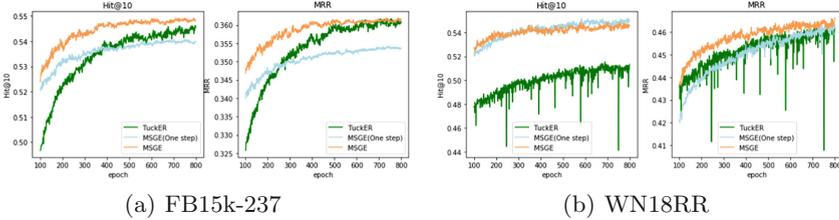
For different datasets, the best setting of the number of iterations varies a lot. For FB15k and UMLS the number at 1 provides the best performance, however for other datasets, iterative mechanism is helpful for boosting the performance. The best number of iterations is set to 5 for WN18, 3 for WN18RR, 8 for FB15k-237 and 2 for Kinship.

### 4.3 Link Prediction Results

We do link prediction task on 6 benchmark datasets, comparing with several classical baselines such as TransE [4], DistMult [10] and some SOTA strong baselines such as ConvE [14], RotatE [8] and TuckER [13]. For smaller datasets

**Table 4.** Influence of number of iterations on FB15k-237.

Length	FB15k-237			
	MRR	Hit@10	Hit@3	Hit@1
1	0.349	0.536	0.384	0.255
3	0.354	0.539	0.389	0.261
5	0.354	0.541	0.391	0.261
8	<b>0.357</b>	<b>0.544</b>	<b>0.392</b>	<b>0.264</b>
10	0.355	0.540	0.391	0.263

**Fig. 4.** Convergence study between Tucker and MSGE(ours) on FB15k-237 and WN18RR.

UMLS and Kinship, we also compare with some non-embedding methods such as NTP [26] and NeuralLP [27] which learn logic rules for predicting, as well as MINERVA [28] which utilizes reinforcement learning for reasoning over paths in knowledge graphs.

The results are reported in Table 2 and Table 3. Overall, from the results we can conclude that our model achieves comparable or better performance than SOTA models on datasets. Even with datasets without inverse relations such as WN18RR, FB15k-237 which are more difficult datasets, our model can still achieve comparable performance.

#### 4.4 Analysis on Number of Iterations

To study the effectiveness of the iterative multi-step architecture, we list the performance of different number of steps on FB15k-237 in Table 4. The model settings are all exactly the same except for length of steps. From the results on FB15k-237 we can conclude that the multi-step mechanism indeed boosts the performance for a complex knowledge graph like FB15k-237, which verify our motivation that refining information for several steps can obtain more helpful information for some complex datasets.

#### 4.5 Convergence Study

We report the convergence process of Tucker and MSGE on FB15k-237 dataset and WN18RR dataset in Fig. 4. We re-run Tucker with exactly the same settings

**Table 5.** Parameter counts comparison.

Dataset	ConvE	TuckER	MSGE
WN18	10.32M	9.39M	8.48M
WN18RR	10.31M	9.39M	8.84M
FB15k	6.16M	11.53M	3.81M
FB15k-237	5.19M	11.00M	3.57M

**Table 6.** Ablation study on FB15k-237.

Model	FB15k-237			
	MRR	Hit@10	Hit@3	Hit@1
MSGE	<b>0.357</b>	<b>0.544</b>	<b>0.392</b>	<b>0.264</b>
No gate	0.301	0.459	0.327	0.222
Concat	0.349	0.534	0.384	0.256
Replicate	0.351	0.537	0.388	0.257

claimed in the paper. All the results stand for the performance on valid dataset. For MSGE, we also report the result of one step for comparison. It is obvious that MSGE can converge rapidly compared to TuckER with nearly the same or better final performance. From the analysis of model architecture, TuckER needs an extra core tensor  $W$  to capture interactive information. While in MSGE, entities and relations are directly interacted with each other through a gate cell. On dataset WN18RR, we can find that the convergence process of TuckER is not as steady as MSGE, which demonstrates the efficiency of our model.

#### 4.6 Efficiency Analysis

In Table 5, we report the parameter counts of ConvE, TuckER and our model for comparison. Our model can achieve better performance on most datasets with much less parameters, which means our model can be more easily migrated to large knowledge graphs. As for TuckER, which is the current SOTA method, the parameter count is mainly due to the core interaction tensor  $W$ , whose size is  $d_e * d_r * d_e$ . As the grow of embedding dimension, this core tensor will lead to a large increasing on parameter size. However, note that our model is an iterative architecture therefore only a very few parameters are needed apart from the embedding, the complexity is  $\mathcal{O}(n_e d + n_r d)$ . For evaluating time efficiency, we re-run TuckER and our model on Tesla K40c. TuckER needs 29 s/28 s to run an epoch on FB15k-237/WN18RR respectively, MSGE needs 17 s/24 s respectively, which demonstrate the time efficiency due to few operations in our model.

## 4.7 Ablation Study

To further demonstrate our motivation that gate mechanism and multi-step reasoning are beneficial for extracting information. We do ablation study with the following settings:

- **No gate:** Remove the gates in our model to verify the necessity of controlling information flow.
- **Concat:** Concatenate information extracted in every step together and feed them into a fully connected layer to obtain another kind of final information, which is used to verify that more useful information are produced by the procedure of multi-step.
- **Replicate:** Replicate the relation to gain  $k$  same query relations for training. This is to prove that extracting diverse information from multi-view query relations is more helpful than using the same relation for  $k$  times.

The experiment results are reported in Table 6. All results demonstrate our motivation that controlling information flow in a multi-step way is beneficial for link prediction task in knowledge graphs. Especially a gated cell is of much benefit for information extraction.

## 5 Conclusion and Future Work

In this paper, we propose a multi-step gated model MSGE for link prediction task in knowledge graph completion. We utilize gate mechanism to control information flow generated by the interaction between subject entity and relation. Then we repeat gated module to refine information from coarse to fine. It has been proved from the empirical results that utilizing gated module for multiple steps is beneficial for extracting more useful information, which can further boost the performance on link prediction. We also do analysis from different views to demonstrate this conclusion. Note that, all information contained in embeddings are learned across the training procedure implicitly. In future work, we would like to aggregate more information for entities to enhance feature extraction, for example, from the neighbor nodes and relations.

## References

1. Bollacker, K.D., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: ACM SIGMOD, pp. 1247–1250 (2008)
2. Mahdisoltani, F., Biega, J., Suchanek, F.M.: Yago3: A knowledge base from multilingual wikipeidias (2015)
3. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). [https://doi.org/10.1007/978-3-540-76298-0\\_52](https://doi.org/10.1007/978-3-540-76298-0_52)
4. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NIPS, pp. 2787–2795 (2013)

5. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. *AAAI* **14**, 1112–1119 (2014)
6. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. *AAAI* **15**, 2181–2187 (2015)
7. Ebisu, T., Ichise, R.: Knowledge graph embedding on a lie group. In: *AAAI*, Toruse (2018)
8. Sun, Z., Deng, Z.H., Nie, J.Y.: Rotate: knowledge graph embedding by relational rotation in complex space. In: *ICLR* (2019)
9. Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: *ICML* (2011)
10. Yang, B., Yih, W., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: *ICLR* (2015)
11. Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., Bouchard, G.: Complex embeddings for simple link prediction. In: *ICML*, pp. 2071–2080 (2016)
12. Kazemi, S.M., Poole, D.: Simple embedding for link prediction in knowledge graphs. In: *NIPS* (2018)
13. Balažević, I., Allen, C., Hospedales, T.: Tensor factorization for knowledge graph completion. In: *EMNLP* (2019)
14. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2D knowledge graph embeddings. In: *AAAI* (2018)
15. Schlichtkrull, M., Bloem, T.N., Kipf, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: *European Semantic Web Conference* (2018)
16. Shen, Y., Huang, P.S., Gao, J. Reasonet: learning to stop reading in machine comprehension. In: *ACM SIGKDD* (2017)
17. Dhingra, B., Liu, H., Yang, Z.: Gated-attention readers for text comprehension (2016)
18. Ji, G., He, S., Xu, L., Liu, K.: Knowledge graph embedding via dynamic mapping matrix. In: *ACL* (2015)
19. Lacroix, T., Usunier, N., Obozinski, G.: Canonical tensor decomposition for knowledge base completion. In: *ICML* (2018)
20. Qian, W., Fu, C., Zhu, Y., Cai, D., He, X.: Translating embeddings for knowledge graph completion with relation attention mechanism. In: *European Semantic Web Conference* (2018)
21. Hochreiter, S., Schmidhuber, J.: Long short-term memory. *Neural Comput.* **9**, 1735–1780 (1997)
22. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint* (2014)
23. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, I., Wojna, Z.: Rethinking the inception architecture for computer vision. In: *Proceedings of IEEE CVPR*, pp. 2818–2826 (2016)
24. Toutanova, K., Chen, D.: Observed versus latent features for knowledge base and text inference. In: *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pp. 57–66 (2015)
25. Kok, S., Domingos, P.: Statistical predicate invention. In: *ICML* (2007)
26. Rocktaschel, T., Riedel, S.: End-to-end differentiable proving. In: *NIPS* (2017)
27. Yang, F., Yang, Z., Cohen, W.: Differentiable learning of logical rules for knowledge base reasoning. In: *NIPS* (2017)
28. Das, R., et al.: Go for a walk and arrive at the answer: reasoning over paths in knowledge bases using reinforcement learning. In: *ICLR* (2018)