TECHNICAL REPORT OF TEAM GRAPHMIRACLES IN THE WIKIKG90M-LSC TRACK OF OGB-LSC @ KDD CUP 2021

A PREPRINT

Jianyu Cai **Jiajun Chen** **Taoxing Pan** **Zhanqiu Zhang** **Jie Wang**[*]
University of Science and Technology of China
{jycai,jiajun98,tx1997,zqzhang}@mail.ustc.edu.cn
jiewangx@ustc.edu.cn

June 23, 2021

ABSTRACT

In this technical report, we introduce the solution of our team GraphMIRAcles in the WikiKG90M-LSC track of OGB-LSC @ KDD Cup 2021. In the WikiKG90M-LSC track, the goal is to automatically predict missing links in WikiKG90M, a large scale knowledge graph extracted from Wikidata. To address this challenge, we propose a framework that integrates three components—a basic model ComplEx-CMRC, a rule miner AMIE 3, and an inference model to predict missing links. Experiments demonstrate that our solution achieves an MRR of 0.9707 on the test dataset. Moreover, as the knowledge distillation in the inference model uses test tail candidates—which are unavailable in practice—we conduct ablation studies on knowledge distillation. Experiments demonstrate that our model without knowledge distillation achieves an MRR of 0.9533 on the full validation dataset.

1 Method

In this part, we introduce our proposed method in detail. In Section 1.1, we introduce the overall architecture of our method. In section 1.2, 1.3, and 1.4, we introduce the three components of our method, respectively.

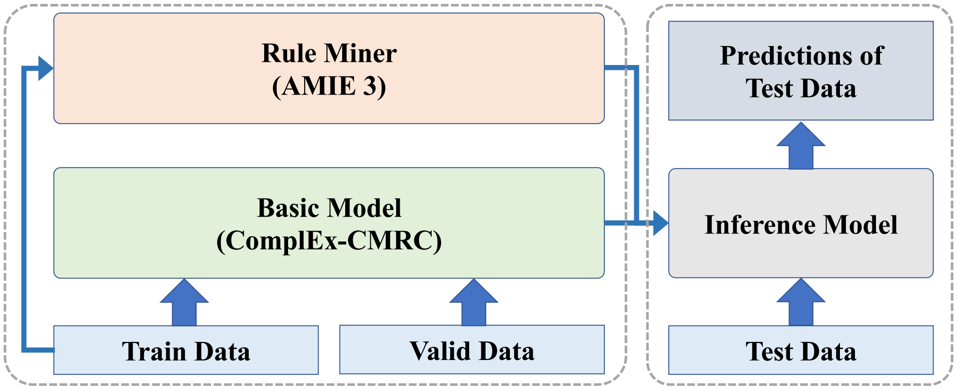

Figure 1: Overall architecture of our method.

[*] Corresponding author.

1.1 Overall Architecture

The overall architecture of our method is shown in Figure 1. Our method contains three components—the basic model ComplEx-CMRC, the rule miner AMIE 3 and the inference model. First, we train the basic models ComplEx-CMRC on the training dataset, and select the best ones based on their performances on the valid dataset. Then, we apply the rule miner AMIE 3 to generate Horn rules based on the training dataset. Finally, based on the basic models and the generated rules, we build an inference model to make predictions on the given test data.

1.2 The ComplEx-CMRC Model

To fully exploit the semantic information embedded in RoBERTa features and the structural information embedded in shallow features, we propose a model named ComplEx-CMRC, in which **CMRC** is our proposed encoder and **ComplEx** is the decoder. "CMRC" is the abbreviation for *Concat-MLP with Residual Connection*.

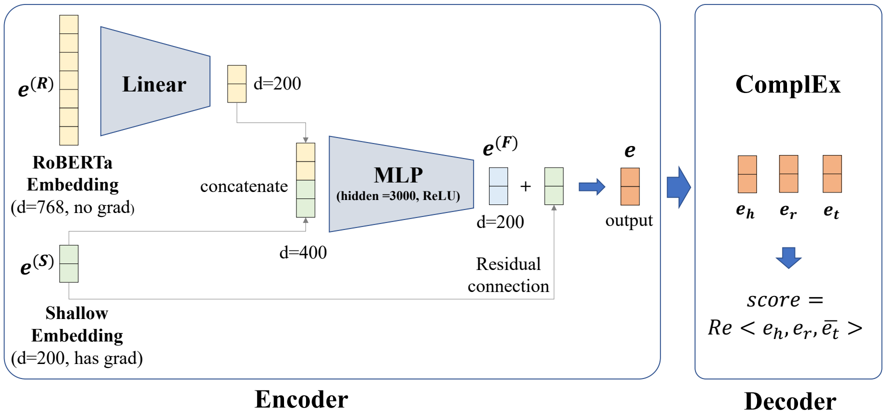

Figure 2: Overview of the ComplEx-CMRC model. ComplEx-CMRC consists of two parts—a CMRC encoder and a ComplEx decoder.

1.2.1 Encoder: Concat-MLP with Residual Connection (CMRC)

In this part, we introduce the proposed encoder CMRC. We use the same encoder architecture but two sets of parameters for entities and relations, respectively. For an arbitrary entity or relation, we are given the RoBERTa embedding $\mathbf{e}^{(R)} \in \mathbb{R}^{768}$, which encodes the semantic information. To capture the structural information in the KG, we define a shallow embedding $\mathbf{e}^{(S)} \in \mathbb{R}^d$ for each entity and relation, where $d < 768$ is the dimension size.

First, we use a linear layer to project $\mathbf{e}^{(R)} \in \mathbb{R}^{768}$ to $\mathbb{R}^d$. Then, we fuse the two kinds of embeddings by first concatenating them and then encoding the concatenated embedding with an MLP. That is, the fused embedding $\mathbf{e}^{(S)} \in \mathbb{R}^d$ is obtained by

$$\mathbf{e}^{(F)} = \text{MLP}([\text{Linear}(\mathbf{e}^{(R)}), \mathbf{e}^{(S)}]), \tag{1}$$

where $[\cdot, \cdot]$ denotes embedding concatenation and $\text{MLP}(\cdot)$ denotes a multi-layer perceptron with one hidden layer.

Finally, we apply residual connection to enable direct gradient flow to the shallow embeddings. That is, the final output of the encoder e is obtained by

$$\mathbf{e} = \mathbf{e}^{(F)} + \alpha \mathbf{e}^{(S)}, \tag{2}$$

where $\alpha \in \mathbb{R}$ is a trainable weight parameter.

1.2.2 Decoder: ComplEx

We choose ComplEx [Trouillon et al., 2016] as the decoder. For a triplet (h, r, t), the encoder generates the embedding of h, r and t, which are $\mathbf{e}_h \in \mathbb{R}$, $\mathbf{e}_r \in \mathbb{R}$, and $\mathbf{e}_t \in \mathbb{R}$, respectively. We then transform those embeddings from $\mathbb{R}^d$ to $\mathbb{C}^{d/2}$ by regarding the first $d/2$ dimensions as the real part and the rest as the imaginary part.

Then, the score $f(h, r, t)$ of (h, r, t) is computed by

$$f(h, r, t) = \text{Re} < \mathbf{e}_h, \mathbf{e}_r, \mathbf{e}_t > . \tag{3}$$

Table 1: The design choices of encoders. In the table, we list four types of encoders.

Encoder	**Description**
Concat	$\mathbf{e} = \text{Linear}([\mathbf{e}^{(R)}, \mathbf{e}^{(S)}])$
Concat-MLP	$\mathbf{e} = \text{MLP}([\text{Linear}(\mathbf{e}^{(R)}), \mathbf{e}^{(S)}])$
Concat-MLP-Residual (w/o weights)	$\mathbf{e} = \text{MLP}([\text{Linear}(\mathbf{e}^{(R)}), \mathbf{e}^{(S)}]) + \mathbf{e}^{S}$
Concat-MLP-Residual	$\mathbf{e} = \text{MLP}([\text{Linear}(\mathbf{e}^{(R)}), \mathbf{e}^{(S)}]) + \alpha\mathbf{e}^{S}$

1.3 Rule Miner

Knowledge graph contains a wealth of structural information, and we can mine *rules* from the knowledge graph. For example, we can mine the rule

$$\text{livesIn}(h,p) \wedge \text{marriedTo}(h,w) \Rightarrow \text{livesIn}(w,p)$$

This rule captures the fact that the spouse of a person usually lives in the same place as the person [Galárraga et al., 2015]. We can acquire such rules from the training data, and use them to enhance the performance of the inference model.

We use the code of AIME 3 to generate rules from the knowledge graph constructed by the training data. As the whole knowledge graph is too large, we sample five subgraphs from the whole graph. Then we apply the AMIE 3 to generate rules from the five subgraphs, respectively. Finally, we merge all the rules to get the final rules.

After getting the generated rules, we use them to make prediction for unseen data. Suppose that the set of entities is $\mathcal{E} = \{e_1, e_2, \ldots, e_{|\mathcal{E}|}\}$ and the set of relations is $\mathcal{R} = \{r_1, r_2, \ldots, r_{|\mathcal{R}|}\}$. We define the adjacent matrix of k-th relation r_k as $\mathbf{M}_{r_k} \in \{0,1\}^{|\mathcal{E}\times\mathcal{E}|}$, where $[\mathbf{M}_{r_k}]_{ij} = 1$ if and only if (e_i, r_k, e_j) is a triple in the knowledge graph. For a rule $r_c(x,y) \Leftarrow r_a(x,y) \wedge r_b(y,z)$, the we can calculate the adjacent matrix of new triples for relation r_c as follows.

$$\mathbf{M}^N_{r_c} = \mathbf{M}_{r_a}\mathbf{M}_{r_b} - \mathbf{M}_{r_c}$$

The matrix $\mathbf{M}^N_{r_c}$ contain the predictions of some unseen triples. Therefore, we can use the generated rules to promote the prediction of new triples. The calculations of other rules can be induced similarly.

1.4 Inference Model

In this part, we introduce the inference model. In Section 1.4.1, we introduce the overall architecture. In Section 1.4.2, we introduce the inference process.

1.4.1 Overall Architecture

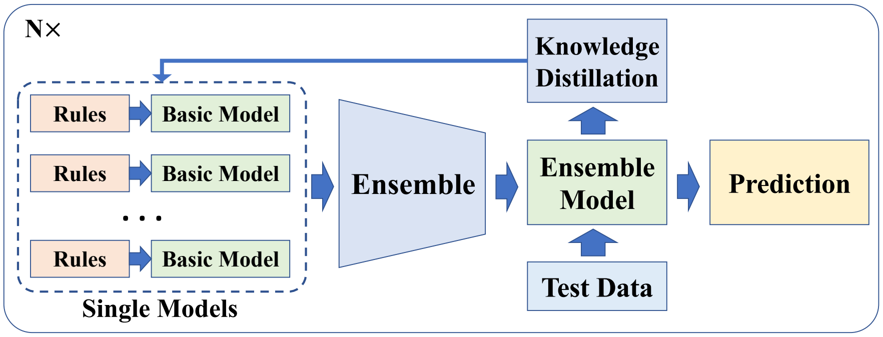

Figure 3: The overall architecture of our inference model.

In this part, we introduce the design of our inference model, the overall architecture is shown in Figure 3. The inference model takes the trained basic models, the mined rules and the test data as input, and it outputs the predictions on the test dataset. To make accurate predictions on the test dataset, the inference applies rule-based data augmentation, ensemble methods and knowledge distillation techniques. In the model, the inference procedure repeats several times, and the final prediction on the test set is the prediction of the last iteration.

1.4.2 The Inference Process

Rule-based Data Augmentation We filter the rules in the KG by their confidence, and use high-confident rules to generate new unseen triples (see Section 1.3). We then use the newly generated triples to finetune the basic model.

Ensemble Given N trained single models, we apply average bagging to obtain a better ensemble model. Let $S_i \in \mathcal{R}^{|\mathcal{D}_{test}| \times n_c}$ denote the prediction of the i-th single model on the test set, where $|\mathcal{D}_{test}|$ denote the number of test samples, and n_c denotes the number of candidate entities of each sample. When making predictions with the ensemble model on test data, the predictions of the ensemble model is

$$S = \frac{\sum_{i=1}^{N} S_i}{N}, \tag{4}$$

where N is the number of single models for ensembling.

Knowledge Distillation Since the ensemble model significantly outperforms single models, we use knowledge distillation [Hinton et al., 2015] to distill the superior performance of the ensemble model into single models. Specifically, we perform knowledge distillation by allowing the single models to learn the output of the ensemble model. In this way, the newly distilled single models could achieve similar performances to the previous ensemble model. We then repeat ensembling and distillation for several times. We use the predictions of the ensemble model at the last iteration as the final predictions of test data. Note that the knowledge distillation uses test tail candidates, which are unavailable in real-world machine learning problems, as models need to rank true tails among all the entities in practice. To evaluate our method in a more practical setting, we report the MRR of our model without knowledge distillation in Section 2.4.

2 Experiments

In Section 2.1, we introduce the training protocols. In Section 2.2, we conduct the ablation studies on model design. In Section 2.3, we present the details of rule mining. In Section 2.4, we show the performance of the inference model on the validation dataset.

2.1 Training Protocols

2.1.1 The Inverse Relation Setting

We adopt the "inverse relation setting" for training. That is, we define an inverse relation r^{-1} for each relation r in the KG. Then, for each triplet (h, r, t) in the training dataset, we add a new triplet (t, r^{-1}, h) to the dataset. During training, each relation r and its inverse relation r^{-1} share the same RoBERTa embedding, but their shallow embeddings are different.

After adding inverse relations, we only keep the "tail mode" during training. That is, for each triplet (h, r, t), we only require the model to predict the tail entities t given the query $(h, r, ?)$. Since inverse relations are introduced, the head prediction task is also included through the triplet (t, r^{-1}, h) built from the inverse relation r^{-1}.

Table 2: The ablation studies on model design.

Decoder	**Encoder**	**InvRel**	**MRR (5% Validation)**
DistMult	Concat	No	0.856
ComplEx	Concat	No	0.852
ComplEx	Concat	Yes	0.887
ComplEx	Concat-MLP	Yes	0.905
ComplEx	Concat-MLP-Residual	Yes	0.926

2.1.2 Hyperparameters

The hyperparameters in our method are as follows.

- The embedding dimension: 300

Table 3: The sampled subgraphs and the number of rules.

Subgraph ID	**Sampled Triples**	**Number of Samples**	**Number of generated rules**
0	train_hrt[0 : 200000000]	200,000,000	7179
1	train_hrt[200000000 : 400000000]	200,000,000	4981
2	train_hrt[400000000 :], train_hrt[: 100000000]	201,160,482	8026
3	train_hrt[100000000 : 300000000]	200,000,000	3903
4	train_hrt[300000000 :]	201,160,482	5999

- The intermediate dimension of MLP: 3000
- Learning rate for shallow embeddings: 1e-1
- Learning rate for MLP parameters: 1e-4
- The number of processes: 4
- Batch size: 800
- Negative sample size: 100

2.2 Ablation Studies on Model Design

In this part, we conduct ablation studies on the model design. The design choices of encoders are listed in Table 1. The results are shown in Table 2. Experiments show that *Concat-MLP-Residual* outperforms other encoders on the validation dataset. In Table 2, InvRel denotes the "inverse relation" setting, which is described in Section 2.1.

2.3 The Generated Rules by AMIE 3

We use AMIE 3 to mine rules from the knowledge graph. For computational efficiency, we sample five subgraphs from the whole graph and only mine rules of length no longer than 3. We show the sampled subgraphs and the number of rules in Table 3, where we use train_hrt to represent the NumPy array of the training triples. After getting the rules from the five subgraphs, we merge all the rules and finally get 11716 rules. We filter the rules by their confidence. For confidence greater than 0.95, there are 2062 rules. For confidence greater than 0.99, there are 1464 rules.

2.4 The Performance of Inference Model

In this part, we conduct experiments on the inference model. We repeat ensembling and distillation for three times. Table 4 shows the performance of the single model and the ensemble model on validation data. We use the ensemble model of Stage 3 to get the final predictions of the test data.

Table 4: The MRR of the single model and the ensemble model on validation data at different inference stage.

Model	Stage 0	Stage 1	Stage 2	Stage 3
Single Model	0.926	0.970	0.973	0.976
Ensemble Model	0.953	0.973	0.977	**0.978**

As discussed in Section 1.4.2, the knowledge distillation process uses the test tail candidates. To evaluate our method in a more practical setting, we conduct ablation studies by excluding the knowledge distillation process, and the results are shown in Table 5. Experiments show that our model achieves an MRR of 0.9533 on the validation dataset without knowledge distillation (i.e., this model does not have any access to the provided test/val tail candidates).

Table 5: The ablation studies on Knowledge Distillation (KD). Note that "Ensemble Model without KD" does not have any access to the provided test/val tail candidates.

Model	MRR (Full Validation)	MRR (Full Test)
Ensemble Model with KD	0.9782	0.9707
Ensemble Model without KD	0.9533	-

3 Conclusion

In this paper, we introduce our proposed method for WikiKG90M-LSC track of KDD Cup 2021. In our method, we integrate three components—the basic model ComplEx-CMRC, the rule miner AMIE 3 and the inference model. In the inference model, we apply knowledge distillation by using test tail candidates, which are unavailable in the practical KG completion scenario. To evaluate our model in a more practical setting, we conduct ablation studies on knowledge distillation, and report the results that does not use val/test tail candidates. Experiments on the link prediction task demonstrate the effectiveness of our proposed method.

References

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

Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M. Suchanek. Fast rule mining in ontological knowledge bases with AMIE+. *VLDB J.*, 24(6):707–730, 2015.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.