Solution for NeurIPS 2022 OGB-LSC

Xu Chen
Microsoft Research Asia
Beijing, China
xu.chen@microsoft.com

Xiaojun Ma
Microsoft Research Asia
Beijing, China
xiaojunma@microsoft.com

Lun Du
Microsoft Research Asia
Beijing, China
lun.du@microsoft.com

Yue Fan *
Peking University
Beijing, China
fanyue@pku.edu.cn

Jiayi Liao*
University of Science and Technology of China
Anhui, China
ljy0ustc@mail.ustc.edu.cn

Chongjian Yue
Peking University
Beijing, China
chongjian.yue@stu.pku.edu.cn

Qiang Fu
Microsoft Research Asia
Beijing, China
qifu@microsoft.com

Shi Han
Microsoft Research Asia
Beijing, China
shihan@microsoft.com

Abstract

In this technical report, we detail our solution to the challenge on WikiKG90Mv2 of the 2nd OGB-LSC [3]. WikiKG90Mv2 is an anonymized knowledge graph consisting of 91,230,610 entities, 1,387 relations and 601,062,811 triplets (edges) extracted from the Wikidata knowledge base. Given a query of a head entity and a relation, the task is to predict its tail entity from the entire set of over 90 million entities. Our solution contains three steps. The first step is to generate a small set of tail candidates for each query using a structure-based strategy and rule mining. In the second step, we train 13 existing knowledge graph embedding models and generate 10 manual features. We then use each model and feature to predict a score for each candidate tail with respect to the query. In the final step, we adopt the ensemble method to assemble the scores given by 13 knowledge graph embedding models and 10 manual features. The MRR on the validation and test-challenge set are 0.2918 and 0.2514, respectively.

1 Introduction

Knowledge graphs, facilitating many downstream tasks such as search question answering and recommendation system, are used to depict relations among entities through graph structures. The edges in the knowledge graph are presented in the form of (h, r, t), where h is the head entity, r is the relation and t is the tail entity. Large-scale knowledge graphs such as Wikidata aim to collect all the relations between entities. However, with the explosion of information in the fast-paced society, it is challenging for the knowledge graph to comprehensively preserve all the facts in time. Therefore, the task to predict unseen facts based on existing facts arouses the interest of researchers. The link-prediction task of OGB-LSC is to predict the unseen tail t given a query (h, r) on the anonymized knowledge graph of Wikidata.

In the 1st OGB-LSC, for every query, a candidate set consisting of 1 ground truth tail and 1000 negative tails. However, for the task of link prediction in the 2nd OGB-LSC, the sets of tail candidate are no longer provided, which brings a huge challenge that the true tail entity need to be selected

*Work performed during the internship at MSRA.

from over 90 million entities. Such settings sharply increase the difficulty of the task. As a result, *how to generate appropriate candidates for each query* becomes the focus of our solution. One assumption is that if we could provide a candidate set that is as good as the one in the 1st OGB-LSC, then applying the winner solutions should achieve similar performance. Regarding the fact that the true tails are usually *close* to the head on the knowledge graph, we design the graph structure-based strategy and take advantage of rule mining [2] to generate the tail candidates for the query. We then train 13 knowledge graph embedding models and generate 10 manual features to score each candidate tail of a given query. We predict the final score by the ensemble of 23 models, and the coefficients of all models are tuned on the validation set. The same coefficients are used to generate the scores on the test-challenge set for the final submission.

2 Candidate Generation

For every query consisting of a head and a relation, we first generate a small tail candidate set from the 90 million entities, which can significantly reduce the computation cost regarding the large entity set. Graph structure play an important role in the Wikidata, which benefit the knowledge graph complementation task. Furthermore, we augment the candidates with mined relations. In conclude, the candidates are generated by both the structure-based strategy (20000 candidates at maximum) and rule mining (50 candidates at maximum) strategy.

2.1 Structure-Based Strategy

The structure-based strategy generates candidates by the graph statistics of the entities. Given a query (h, r_i), with r_i being the i-th relation, the tail candidates are those entities that have co-occurrence with relation r_i with large in-degrees.

For every relation r_i, we first create its tail set as $D_i = \{t | \exists h, (h, r_i, t) \in \mathcal{K}\}$, where $\mathcal{K}$ is the union of the training set and the validation set. We also calculate the in-degrees of all entities. For every query (h, r_i), we choose top-$min(s, |D_i|)$ entities from D_i with the highest in-degree as the structure-based candidates for the query, where s represents the candidate size.

Table 1: The hit counts and hit rates of the structure-based candidates on the validation set.

Candidate Size	**1000**	**2000**	**5000**	**10000**	**20000**	**30000**	**18880842**
Hit Count	6545	7082	7805	8243	8654	8868	10080
Hit Rate	43.63%	47.21%	52.03%	54.95%	57.69%	59.12%	67.20%

For a query and its ground-truth tail, if the true tail is among the candidates of this query, we call it a hit. Table 1 shows the hit rate of the structure candidates on the validation set which contains 15000 queries. The hit rate increases as the candidate size grows, and it is $\max_i |D_i| = 18880842$ at most. A trade-off between a higher hit rate and a smaller candidate size (meaning less computation cost) requires careful consideration, and we finally set the candidate size to 20000 for the structure-based strategy.

2.2 Rule Mining

Rule Mining is a technique for finding hidden rules, which can be used to predict unseen triplets in the knowledge base. For instance, the following rule holds for most situations:

$$motherOf(x, z) \wedge marriedTo(x, y) \Rightarrow fatherOf(y, z),$$

which means that if x is the mother of z and x is married to y, then y is the father of z. Here x, y and z are variables of entities.

We first generate five sub-graphs from the original training set and then use AMIE [2] to mine rules. On each sub-graph, rules are mined under different PCA confidence (confidence under the Partial Completeness Assumption) thresholds. we combine all the rules from the five sub-graphs and generate new triplets based on the whole training set and these rules. For every query, the tails generated by rule mining serve as the rule-mining candidates. For most queries, rule-mining generates 0 or 1 candidate, therefore we treat the rule-mining candidates as complementary to the structure-based candidates.

Table 2: The hit counts of the rule-mining candidates on the validation set under different PCA confidence thresholds and candidate sizes.

Hit Count	**Size=10**	**Size=20**	**Size=50**
conf>=95%	448	450	451
conf>=80%	899	906	912

Table 2 presents the hits of the rule-mining candidates on the validation set. It can be found that setting the PCA confidence threshold to 80% and candidate size to 50 results in a better hit rate, which is the setting we adopt to generate the rule-mining candidates. It should be pointed out that for a query having a large number of rule-mining tails, we randomly select 50 tails as its candidates. By combining the rule-mining candidates with the structure-based candidates, the final number of hits on the 15000 validation query is 8930, corresponding to a hit rate of 59.53%.

3 Scoring Methods

Given a query containing the head h and relation r, and the corresponding candidate set $\mathcal{T}$, the scoring methods aim to measure the probability of each candidate $t \in \mathcal{T}$ to be the real tail. Here we utilize two kinds of scoring methods, knowledge graph embedding methods and feature-based scoring.

3.1 Knowledge Graph Embedding Methods

From the winning solution of 1st LSC@KDD Cup 2021 we discover that traditional knowledge graph embedding (KGE) methods work well in this challenge. So we select 13 different KGE methods, and the details are organized in Table. 3:

Table 3: KGE results on generated candidates

Model name	Dim	B	lr	γ	reg_coef	#Negs	Seed	Max step	Best step	MRR(val)
AutoSF [9]	768	2000	0.15	50	1e-6	2000	0	10M	6.875M	0.1819
AutoSF_concat	768	2000	0.15	50	1e-6	2000	0	10M	0.725M	0.2006
ComplEx_c [7]	512	2000	0.1	10	1e-9	2000	0	1.5M	1.5M	0.2019
ComplEx_d	512	2000	0.1	10	1e-9	2000	1	1M	0.98M	0.1941
ComplEx_e	512	2000	0.1	10	1e-9	2000	9	1M	1M	0.1963
ComplEx_f	512	2000	0.1	10	1e-9	2000	77	10M	2.875M	0.2086
ComplEx_shallow	512	2000	0.1	10	1e-9	2000	77	1M	0.85M	0.1764
DistMult_g [8]	512	2000	0.1	10	1e-9	2000	13	1M	0.975M	0.1986
DistMult_shallow	512	2000	0.1	10	1e-9	2000	13	10M	2.6M	0.1705
RotatE_concat [6]	100	1000	0.1	8	1e-9	1000	0	10M	1.6M	0.1707
RotatE_shallow	256	1024	0.1	12	1e-9	1024	0	10M	7.85M	0.2008
SimplE_i [4]	512	2000	0.1	10	1e-9	2000	77	1M	1M	0.1886
TransE_a [1]	768	2000	0.2	10	1e-9	2000	0	10M	9.95M	0.2277

Here "_concat" refers to concatenating shallow embeddings and MPNet embeddings and then feeding them into a two-layer MLP. 'B' denotes the batch size, 'reg_conf' is the regularization coefficient, and '#Negs' is the number of negative samples. More descriptions about the hyper-parameters can be found in the document of DGL-KE. MRR here is calculated based on the candidate set of Smore. Note that the candidate set $\mathcal{T}$ may vary when the optimal candidate generation strategy is not determined. To reduce the training time, we will save model parameters of the best MRR metric. Hence, we can compute the score of a candidate t given (h, r) with the score function of each KGE method, e.g., $|E_h + E_r - E_t|$ of TransE. E_* is the learned embedding for entity/relation $*$.

3.2 Feature-base scoring

In addition to trainable models, manual feature engineering[2] also play an important role in our work. We adopt the design of the first winner solution in KDD Cup 2021 WikiKG90M-LSC. The final

[2]https://ogb.stanford.edu/paper/kddcup2021/wikikg90m_BD-PGL.pdf

task is to predict the tail given head and relation, therefore the manual features include **Head-to-tail Features** and **Relation-to-tail Features**.

3.2.1 Head-to-tail Features

Head to Tail features predict the probability of reaching tails from the current head. For a head, relation and tail triple, there are six different walking directions dir, including head to tail (HT), head to relation (HR), relation to head (RH), relation to tail (RT), tail to head (TH) and tail to relation (TR). The probability of e_1 to e_2 in direction dir is defined as follow:

$$P_{\text{dir}}\left(e_1, e_2\right) = \frac{S_{\text{dir}}\left(e_1, e_2\right)}{\sum_{e \sim N_{\text{dir}}\left(e_1\right)} S_{\text{dir}}\left(e_1, e\right)}, \tag{1}$$

where S_{dir} is the frequency of e_1 to e_2 in the direction of dir.

It is worth noting that different from the link prediction in OGB-LSC 2021, no candidate sets are given in OGB-LSC 2022. We caculated all features for the generated validation candidates and test-challenge candidates in section 2. We search for the best weight of our developed models and apply the same rule for final test prediction. Test data is only touched during inference. We take six manual head to tail features as below:

$$\begin{aligned}
F_{HT}(h,t) &= P_{HT}(h,t) \\
F_{TH-HT}(h,t) &= \sum_{e \sim N_{TH}(h) \cap N_{TH}(t)} P_{TH}(h,e) * P_{HT}(e,t) \\
F_{HT-HT}(h,t) &= \sum_{e \sim N_{HT}(h) \cap N_{TH}(t)} P_{HT}(h,e) * P_{HT}(e,t) \\
F_{HT-TH}(h,t) &= \sum_{e \sim N_{HT}(h) \cap N_{HT}(t)} P_{HT}(h,e) * P_{TH}(e,t) \\
F_{TH-TH}(h,t) &= \sum_{e \sim N_{TH}(h) \cap N_{HT}(t)} P_{TH}(h,e) * P_{TH}(e,t) \\
F_{HT-HT-TH}(h,t) &= \sum_{e_1 \sim N_{HT}(h)} \sum_{e_2 \sim N_{HT}(t)} P_{HT}\left(h, e_1\right) * P_{HT}\left(e_1, e_2\right) * P_{TH}\left(e_2, t\right).
\end{aligned} \tag{2}$$

3.2.2 Relation-to-tail Features

We also calculate four relation-to-tail features as follow:

$$\begin{aligned}
F_{RT}(r,t) &= P_{RT}(r,t) \\
F_{RT-TR-RT}(r,t) &= \sum_{e_1 \sim N_{RT}(r)} \sum_{e_2 \sim N_{TR}(t)} P_{RT}\left(r, e_1\right) * P_{TR}\left(e_1, e_2\right) * P_{RT}\left(e_2, t\right) \\
F_{RH-HR-RT}(r,t) &= \sum_{e_1 \sim N_{RH}(r)} \sum_{e_2 \sim N_{HR}(t)} P_{RH}\left(r, e_1\right) * P_{HR}\left(e_1, e_2\right) * P_{RT}\left(e_2, t\right) \\
F_{RT-HR-RT}(r,t) &= \sum_{e_1 \sim N_{RT}(r)} \sum_{e_2 \sim N_{HR}(t)} P_{RT}\left(r, e_1\right) * P_{HR}\left(e_1, e_2\right) * P_{RT}\left(e_2, t\right)
\end{aligned} \tag{3}$$

4 Ensemble

We now have 13 scores from KGE methods and 10 scores from feature-based scoring. Then it is natural to take each score as a weak learner and assemble them for better performance. The scores generated by different methods are highly diverse, so min-max normalization is applied to scale them to [0,1][3]. Based on the scaled scores, we assign a coefficient to each type of score and there are 23 coefficients in total. We use nni [5] to search for an optimal combination. The coefficient search range is [0,1] with TPE tuner. The results are organized in Table. 4 and Table. 5 for reproducibility.

[3]Considering the difference between validation set distribution and test-challenge set distribution, the maximum value and minimum value are generated from the fused data of the validation set and test-challenge set.

Table 4: Coefficient of KGE methods.

Scoring method	Coefficient
AutoSF	0.42621964
AutoSF_concat	0.861696884
ComplEx_c	0.648180783
ComplEx_d	0.999374295
ComplEx_e	0.501315892
ComplEx_f	0.999292391
ComplEx_shallow	0.936766832
DistMult_g	0.82231182
DistMult_shallow	0.495921709
RotatE_concat	0.309670717
RotatE_shallow	0.306348037
SimpIE_i	0.061835772
TransE_a	0.953984436

Table 5: Coefficient of feature-based scoring.

Scoring method	Coefficient
HT_HT	0.271609321
HT_HT_TH	0.643873637
RT	0.135509344
RH_HR_RT	0.020741257
RT	0.399658646
HT_TH	0.827876436
HT	0.436042954
RT_HR_RT	0.141566342
RT_TR_RT	0.132583523
TH_HT	0.607002105

The final score of each tail t can be formulated as:

$$\text{Score}(t) = \sum_{i=1}^{23} C_i * \text{Score}_i(t), \tag{4}$$

For the i-th scoring method, C_i is the corresponding coefficient and $\text{Score}_i(t)$ is the score generated for candidate tail t. By sorting $\text{Score}(t)$ and selecting top-10 results we can obtain the submission answer. Finally MRR is 0.2918 on the validation set and 0.2514 on the test-challenge set.

5 Conclusion

In this report we present our solution from the perspectives of candidate generation, scoring methods and ensemble. Regarding the hit rate around 60% and the gap between MRR and hit rate, there remains room for improvement. Besides, we find more than 10 repeat queries in the validation/test-challenge set, which make the extra queries useless during the evaluation. Maybe carefully selecting the testing set is also worth considering in the next OGB-LSC.

References

[1] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[2] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*, pages 413–422, 2013.

[3] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv preprint arXiv:2103.09430*, 2021.

[4] Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. *Advances in neural information processing systems*, 31, 2018.

[5] Microsoft. Neural Network Intelligence, 1 2021.

[6] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv preprint arXiv:1902.10197*, 2019.

[7] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International conference on machine learning*, pages 2071–2080. PMLR, 2016.

[8] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

[9] Yongqi Zhang, Quanming Yao, Wenyuan Dai, and Lei Chen. Autosf: Searching scoring functions for knowledge graph embedding. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 433–444. IEEE, 2020.