STARGRAPH: A COARSE-TO-FINE REPRESENTATION METHOD FOR LARGE-SCALE KNOWLEDGE GRAPH

Hongzhu Li, Xiangrui Gao, Yafeng Deng†
Qihoo 360 AI Research
{lihongzhu, gaoxiangrui, dengyafeng}@360.cn

ABSTRACT

Conventional representation learning algorithms for knowledge graphs (KG) map each entity to a unique embedding vector, ignoring the rich information contained in neighbor entities. We propose a method named StarGraph, which gives a novel way to utilize the neighborhood information for large-scale knowledge graphs to get better entity representations. The core idea is to divide the neighborhood information into different levels for sampling and processing, where the generalized coarse-grained information and unique fine-grained information are combined to generate an efficient subgraph for each node. In addition, a self-attention network is proposed to process the subgraphs and get the entity representations, which are used to replace the entity embeddings in conventional methods. The proposed method achieves the best results on the ogbl-wikikg2 dataset, which validates the effectiveness of it. The code is now available at https://github.com/hzli-ucas/StarGraph.

1 Introduction

A Knowledge Graph (KG) is a directed graph with real-world entities as nodes and relationships between entities as edges. In this graph, each directed edge together with its head and tail entities form a triple (head entity, relation, tail entity), indicating that the head and tail entities are connected by a relation.

Knowledge graph embedding (KGE), also known as knowledge representation learning (KRL), aims to embed entities and relations into low-dimensional continuous vector spaces to characterize their latent semantic features. A scoring function is defined to measure the plausibility for triples in such spaces. The embeddings of entities and relations are then learned by maximizing the total plausibility of the observed triples. These learned embeddings can be used to implement various tasks such as knowledge graph completion [1, 2], relationship extraction [3], entity classification [4], etc. The plausibility of each triple is calculated on the embeddings of the entities and relations in it, and the embeddings are directly taken out from the embedding tables. Such a shallow lookup decides that those models are inherently transductive. Moreover, the rich information contained in the neighboring triples is not taken into account.

Compared with shallow embedding models, neighborhood neural encoders, that are able to extract structural information, usually perform much better across various KG datasets. Any generic graph neural networks (GNN), e.g. GCN [5], GraphSAGE [6] and GAT [7], could be employed as the encoder. However, there is a problem adopting these methods to large-scale knowledge graphs, for previous graph networks takes the multi-hop subgraph of the node as input. Due to the large number of nodes and edges, multi-hop subgraphs in large-scale graphs can easily exceed the size limit, and the generation of subgraphs and network calculations can be very time-consuming.

The neighborhood can provide extra information for each node so as to obtain a better representation. In order to adopt neighborhood neural encoders in large-scale KG, an intuitive idea is to utilize partial neighborhood information instead of the full multi-hop subgraph. In this paper, we propose to divide the neighborhood information into different levels for consideration, using coarse-grained information to enhance the generalization, and fine-grained information to ensure the distinction of different entity representations. Additionally, in order to reasonably model the subgraph structure
Figure 1: Illustration of a subgraph generated by StarGraph. Dots and lines represent nodes and edges in the graph, respectively, with larger dots indicating anchors. The nodes in different levels of the subgraph are marked with different colors.

and enable sufficient interaction of node embeddings, we take the lead in using a self-attentive network to extract the neighborhood information and obtain entity representations.

The coarse-grained representation can be implemented by the anchor-based strategy [8], where a small number of nodes are selected as anchors, and the nearest anchors are sampled into the current subgraph. Finer-grained information can then be provided by part of the neighbors and the current node itself. Figure 1 shows a subgraph consisting of nodes with multi-level information, where the current node is circled in blue, the neighboring nodes in green, and the nearest anchors in red. It can be seen that different nodes may share the same set of anchors, e.g. N1 and N2 in the graph, so the fine-grained nodes sampling is necessary.

The coarse-to-fine subgraph sampling is like how to locate a star in the sky, a few bright stars (anchors) are used to roughly indicate the range, and then neighboring stars are used to pinpoint the location. For subgraphs in the entire graph looks like constellations among the stars, we call the proposed method StarGraph.

2 Related Work

Distance-based models consist a major branch of knowledge graph embedding methods. TransE [1] embeds relations and entities into the same vector space, and the relation embedding is interpreted as a translation from the embeddings of the head entity to the tail entity. It is a simple and effective model, but cannot deal with complex and symmetry relations. RotatE [9] takes inspiration from Euler’s formula and uses the rotations of vectors to explain various relations, where the inverse relations can be modeled by complex embeddings. PairRE [10] propose to learn two embeddings for each relation, respectively used to map head and tail entity embeddings into the corresponding relation space. It extends the model ability to handle sub-relations. TripleRE[11] learns another embedding for each relation on the basis of PairRE, and the extra relation embedding is used as a translation vector between the mapped entity embeddings of head and tail. This minor modification significantly improves the model results.

NodePiece [8] borrows the idea of subword tokenization from natural language processing (NLP) and proposes to use a small number of nodes as anchors, constructing vocabulary for each entity using anchors and known relationship types, and using a multi-layer perceptron to obtain node representations calculated from token sequences. However,
there are some drawbacks in this method. It treats tokens in a sequential manner, and the order of tokens introduces noise information to the unordered graph structure. The anchors-based information is coarse and cannot provide good differentiation for nearby nodes. In addition, anchors are used to replace nodes for the purpose of reducing the number of parameters, and have not been considered for information enhancement to improve the model effect.

3 Methodology

The StarGraph is a method to obtain entity representations based on coarse-to-fine neighborhood information. In terms of procedure, it can be divided into two stages, generating and encoding subgraphs. The obtained entity representations, together with the relational embeddings, are used to compute the scores of the triples by a distance-based score function, and optimized with the self-adverse negative sampling loss [9]. We will describe each part in detail below.

3.1 Subgraph Generation

For knowledge graph representation learning, the neighborhood information is proved helpful, and the multi-hop neighbors are often used to enrich the target node description. When the graph is very large and contain enormous nodes, the multi-hop subgraph of the target node could be extremely large and contains much redundant information. We propose to sample a smaller yet more effective subgraph, by describing a node in a coarse-to-fine way. Each subgraph contains three types of nodes as follows.

**Anchors** At first, a minority of nodes are selected as anchors. In order to provide common information, anchors should be well known to different nodes, therefore are chosen according to the node degrees in a decreasing order. For each node, a fixed number of anchors are sampled from its multi-hop neighbors by breadth-first-searching. The number of anchors is a constant value chosen with experience.

**Neighbors** Neighbors refer to a fixed number of nodes sampled from one-hop neighborhood of the target node. Compared with anchors, the neighbors are usually closer to the target node, and can be any node rather than the previously selected minority. The neighbors provide more specificity information than the anchors, but also aim to make a common description for the target node. So we sample the neighbors in a degree-decreasing order.

**Center** To ensure the subgraphs of different nodes being distinguishable, we take the target node itself as the center of its subgraph.

During the generation of subgraphs, anchors are picked from the predefined anchor set, while neighbors and centers could be any node in the total graph. For ease of description, we denote the anchor set as $A$, all nodes of the total graph as $N$. There are $A \in N$, and $|A| < |N|$.

3.2 Encoder

Transformer [12] is proposed by Google in 2017, and widely used in many fields, such as NLP, CV and even graphs. The basic unit in Transformer, the self-attention module provides a novel way to deal with the inputs. It takes a set of features as inputs, update each feature with all other features within the set, and control the information flow with an attention matrix. Compared with multi-layer perceptron (MLP), self-attention module provide a more effective way for the nodes to exchange information and allow the nodes staying unordered. Therefore we adopt self-attention module to build the encoder.

The encoder takes the embeddings of nodes within the subgraph as inputs, each node treated as a token. The nodes in the subgraph are unordered, so the position embedding is not used. Type embedding is added to the token embedding before the transformer layer, indicating each node within the subgraph to be the type of anchor, neighbor or center.

As described in section 3.1, each subgraph consist of nodes from $A$ and $N$, providing coarse and fine-grained information respectively for the target node. As with the entire graph, each node keeps some unique information, while only a small group of nodes are anchors, carrying much general information. Denoting the embedding sizes of $A$ and $N$ as $D_A$ and $D_N$, we set a large $D_A$ to improve the information capacity, and a small $D_N$ to reduce the probability of overfitting. To get the tokens of the same embedding size, $D_A$ is used as the hidden size for transformer layer, and the node embeddings are mapped to dimension $D_A$ by a linear layer ahead.

Since the self-attention layer has a global receptive field, a single layer can adequately mix the information of all nodes. In addition, deep self-attention networks take large scale data and long time pre-training to achieve good results. Therefore, we use a single transformer block, i.e. one attention layer followed by two linear layers, to construct the
network. After the transformer block, the output node features are aggregated by mean-pooling across the subgraph to get the representation of target node.

3.3 Score Function

We adopt a score function that is modified upon TripleRE [11], a recently proposed distance-based method. The formula of TripleREv2 is

\[ f_r(h, t) = -||h \circ (r^h + u \cdot e) - t \circ (r^t + u \cdot e) + r||, \]

where \( \circ \) denotes the element-wise product, \( h \) and \( t \) correspond to the head and tail entity embeddings respectively, \([r^h, r^t, r]\) comprise the relation embedding, \( u \) is a constant specified by users, set to 1 in the original work. We modify equation (1) by switching \( e \) with \( r^h/r^t \), so that the coefficient \( u \) acts on the relation embedding. Now the score of triplets \((h, r, t)\) is calculated as

\[ f_r(h, t) = -||h - t + u \cdot (h \circ r^h - t \circ r^t)||, \]

which can be interpreted as a boosting method based on TransE and PairRE. The new form is referred to as TripleRE’ in the rest of this paper. Although TripleRE’ is exactly the same as TripleREv2 when \( u \) equals 1, it consistently performs better during tuning the hyper-parameter \( u \) according to our experimental results.

3.4 Optimization

We utilize the self-adversarial negative sampling loss [9] as objective for training,

\[ L = -\log \sigma(\gamma - f_r(h, t)) - \sum_{i=1}^{n} w_i \log \sigma(f_r(h_i, t_i) - \gamma), \]

\[ w_i = \frac{\exp f_r(h_i, t_i)}{\sum_{j=1}^{n} \exp f_r(h_j, t_j)}. \]

where \( \gamma \) is a fixed margin and \( \sigma \) is the sigmoid function. We randomly sample \( n \) negative triples for each positive one, and \((h_i, r, t_i)\) is the \( i \)-th negative triple.

4 Experiments

4.1 Dataset and Metric

ogbl-wikikg2 To explore the effect of StarGraph on large-scale knowledge graph, we conduct experiments on the ogbl-wikikg2 dataset, which is a knowledge graph extracted from the Wikidata knowledge base [13]. It contains 2,500,604 entities, 535 relation types and 17,137,181 triplet edges (head, relation, tail).

Task and Metric The task is to predict new triplet edges given the training edges. The evaluation metric follows the standard filtered metric widely used in KG. Specifically, each test triplet edges are corrupted by replacing its head or tail with randomly-sampled 1,000 negative entities, while ensuring the resulting triplets do not appear in KG. The goal is to rank the true head (or tail) entities higher than the negative entities, which is measured by Mean Reciprocal Rank (MRR).

4.2 Detailed Settings

We set batch-size to 512, negative-sampling-size to 64, and train the model for 500,000 steps. The learning rate is initialized to 0.0005 and decreased by 0.1 at half the maximum training step. Dropout is adopted after each linear layer, and the drop-ratio is 0.05. AdamW is used as the optimizer in our experiments.

Hyper-parameter \( u \) in equation (2) is tuned and set to 0.1 according to model performance on validation set, and \( \gamma \) in equation (3) is set to 6.0 following PairRE [10].

Unless otherwise stated, we adopt self-attention network as the encoder, and set \( D_A = 256, D_N = 32 \). The number of anchors and neighbors sampled for each subgraph is fixed as 20 and 5 respectively.
Table 1: The results on ogbl-wikikg2 with different encoders. Subgraphs only consist of anchors in this experiment.

| Encoder  | #Dims | Test MRR | Valid MRR |
|----------|-------|----------|-----------|
| MLP      | 256   | 0.6853   | 0.6897    |
| Attention| 256   | 0.6992   | 0.7062    |

Table 2: The results on ogbl-wikikg2 with different subgraphs. Numbers in the brackets indicate how many such type of node are contained in each subgraph. The ✓ indicates that subgraphs contain corresponding type of nodes.

| anchors(20) | neighbors(5) | center(1) | Test MRR | Valid MRR |
|-------------|--------------|-----------|----------|-----------|
| ✓           | ✓            | ✓         | 0.6992   | 0.7062    |
| ✓           | ✓            | ✓         | 0.7150   | 0.7258    |
| ✓           | ✓            | ✓         | 0.7182   | 0.7262    |
| ✓           | ✓            | ✓         | 0.7205   | 0.7291    |

4.3 Ablation Study

We conduct some comparison experiments to explore the effectiveness of the proposed method, including the network architecture and subgraph generation strategy.

**Encoder** To test the effectiveness of self-attention network as the encoder, we use the two-layer MLP adopted in NodePiece as a control. In this group of experiments, networks take only anchors in each subgraph as inputs, abandoning neighbors and center for simplicity. The experiment results are given in table 1. Attention-based network achieved a significant improvement in evaluation metrics over MLP, and is used as the encoder in the following experiments.

**Subgraph** We carry out an ablation study on subgraph generation, where we analyze the performance when we omit neighbors, centers or both types from subgraphs. Table 2 shows that on the basis of anchors, additional node sampling for subgraph significantly improves the results, which validate the effectiveness of the proposed subgraph generation method. Note that anchors+center introduces only one more node into each subgraph while there are five more nodes for neighbors, but anchors+centers performs better than anchors+neighbors. This suggests the importance of subgraph containing more specific information. The combination of the three kinds of nodes achieves the best results, and removing either neighbors or center leads to a drop in results. This proves the superiority of the proposed subgraph generation method which aggregates multi-level information from coarse to fine.

4.4 Results on ogbl-wikikg2

The results are shown in Table 3. Our method achieves 0.7288 (validation set) and 0.7201 (test set) on MRR, that are state-of-the-art results on the ogbl-wikikg2 dataset.

5 Conclusion

In this paper, we propose StarGraph, a novel node representation method that is based on neighborhood information and applicable to large knowledge graphs. The underlying idea of the method is to process the neighborhood information in a hierarchical manner, by generating a subgraph containing three types of nodes to capture the neighborhood information, and employing a self-attentive network to process the information. We achieves best results on ogbl-wikikg2 dataset, which demonstrates the effectiveness of the proposed method. Although StarGraph makes considerable progress compared to previous methods, there is still much room for improvement in it, including but not limited to exploring better subgraph generation strategies and network structures. We also consider applying StarGraph to other related tasks, such as inductive link prediction, relationship prediction and node classification. These are treated as future work.

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] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 28, 2014.
Table 3: The results on ogbl-wikikg2.

| Model           | #Dims  | Test MRR       | Valid MRR      |
|-----------------|--------|----------------|----------------|
| TransE          | 500    | 0.4256±0.0030  | 0.4272±0.0030  |
| RotatE          | 250    | 0.4332±0.0025  | 0.4353±0.0028  |
| PairRE          | 200    | 0.5208±0.0027  | 0.5423±0.0020  |
| AutoSF          | -      | 0.5458±0.0052  | 0.5510±0.0063  |
| ComplEx         | 250    | 0.5027±0.0027  | 0.3759±0.0016  |
| TripleRE        | 200    | 0.5794±0.0020  | 0.6045±0.0024  |
| TripleREv2      | 200    | 0.6045±0.0017  | 0.6117±0.0008  |
| ComplEx-RP      | 50     | 0.6392±0.0045  | 0.6561±0.0070  |
| AutoSF + NodePiece | -    | 0.5703±0.0035  | 0.5806±0.0047  |
| TripleREv2 + NodePiece | 200 | 0.6582±0.0020  | 0.6616±0.0018  |
| TripleREv3 + NodePiece | 200 | 0.6866±0.0014  | 0.6955±0.0008  |
| InterHT + NodePiece | 200 | 0.6779±0.0018  | 0.6893±0.0015  |
| TranS + NodePiece | 200 | 0.6882±0.0019  | 0.6988±0.0006  |
| TranS(large) + NodePiece | -  | 0.6939±0.0011  | 0.7058±0.0018  |
| TripleRE’ + StarGraph | 256/32 | 0.7201±0.0011  | 0.7288±0.0008  |

[3] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 74–84, 2013.

[4] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In Icml, 2011.

[5] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In International Conference on Learning Representations (ICLR), 2017.

[6] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. Advances in neural information processing systems, 30, 2017.

[7] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. Learning attention-based embeddings for relation prediction in knowledge graphs. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4710–4723, 2019.

[8] Mikhail Galkin, Etienne Denis, Jiapeng Wu, and William L. Hamilton. Nodepiece: Compositional and parameter-efficient representations of large knowledge graphs. In International Conference on Learning Representations, 2022.

[9] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In International Conference on Learning Representations, 2018.

[10] Linlin Chao, Jianshan He, Taifeng Wang, and Wei Chu. Pairre: Knowledge graph embeddings via paired relation vectors. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 4360–4369, 2021.

[11] Long Yu, ZhiCong Luo, Deng Lin, HongZhu Li, HuanYong Liu, and YaFeng Deng. Triplere: Knowledge graph embeddings via triple relation vectors. viXra preprint viXra:2112.0095, 2021.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

[13] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. Communications of the ACM, 57(10):78–85, 2014.