Analogy-triple Enhanced Fine-grained Transformer for Sparse Knowledge Graph Completion

Shaofei Wang, Siying Li, and Lei Zou

Peking University, Beijing, China
{wangshaofei,lisiying,zoulei}@pku.edu.cn

Abstract. Sparse problem is the major challenge in knowledge graph completion. However, existing knowledge graph completion methods utilize entity as the basic granularity, and face the semantic under-transfer problem. In this paper, we propose an analogy-triple enhanced fine-grained sequence-to-sequence model for sparse knowledge graph completion. Specifically, the entities are first split into different levels of granularity, such as sub-entity, word, and sub-word. Then we extract a set of analogy-triples for each entity-relation pair. Furthermore, our model encodes and integrates the analogy-triples and entity-relation pairs, and finally predicts the sequence of missing entities. Experimental results on multiple knowledge graphs show that the proposed model can achieve better performance than existing methods, especially on sparse entities.

Keywords: Knowledge graph completion · Sequence-to-sequence model · Sparse knowledge graph

1 Introduction

Knowledge graphs (KGs) contain massive real world knowledge, and the knowledge in KGs is generally represented by structured triples in the form of (h, r, t), where h and t are head and tail entities respectively, and r denotes the relation between h and t. Due to the incompleteness of KGs, various knowledge graph completion methods are proposed to predict missing links based on existing data of KGs. However, as the long-tail distribution of entities in KGs, the sparse problem of entities is inevitable and becomes the major challenge for knowledge graph completion [27]. For example, in the open source KG Freebase [2], up to 58.2% of the entities appear lower than 10 times [6].

Previous knowledge graph completion methods can be divided into two streams. 1) One is embedding-based methods, these methods learn low-dimensional embeddings for entities and relations, and then score the candidate triples based on embeddings [3, 17, 28]. 2) The other line is rule-based methods, which learn logical rules form KGs, and then apply the rules on existing data to predict new triples [11, 29]. However, although much remarkable progress has been achieved, existing methods still face the following **Semantic Under-transfer Problem**:

Semantic Under-transfer Problem. Current methods utilize entities as the basic granularity, and the granularity of entities is too coarse to transfer the semantics well (i.e., under-transfer problem), especially for sparse entities. For example, in Fig.1, the entity `melbourne institute of technology` is a sparse entity, and it is difficult to predict its `location` and `field`. However, if the entity is split into the fine-grained components `melbourne` and `institute of technology`, whose semantics can be transferred from similar entities. As a consequence, the `location` and `field` can be predicted through learning from other analogy-triples (such as (`melbourne park, location, melbourne`) and (`massachusetts institute of technology, field, education`)).

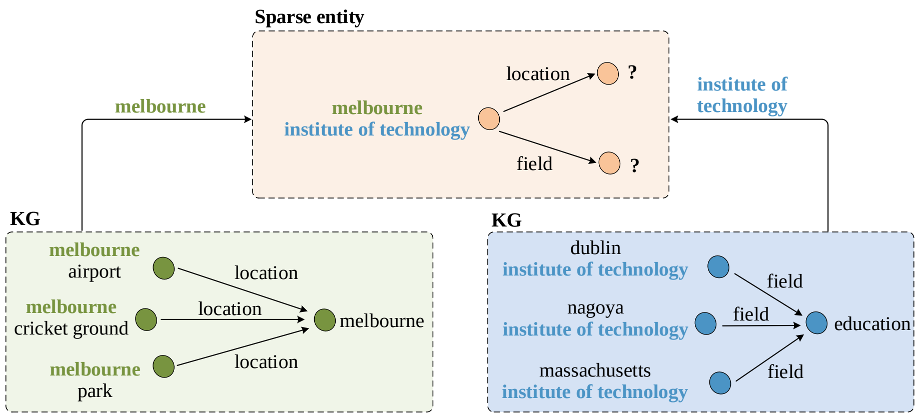

Fig. 1. A motivation example of our knowledge graph completion model on sparse entities. Considering a sparse entity `melbourne institute of technology`, the semantics of this entity is difficult to be modeled by traditional methods due to the data scarcity. While in our method, the entity is split into multiple fine-grained components (such as `melbourne` and `institute of technology`). Thus the semantics of these fine-grained components can be learnt from analogy-triples (showed in the left and right boxes respectively). Finally the `location` and `field` of the sparse entity `melbourne institute of technology` can be predicted.

In this paper, we propose an analogy-triple enhanced fine-grained knowledge graph completion model, the FineKGC, to alleviate the knowledge under-transfer problem. The main motivation of our model can be included as: 1) Traditional models are mainly data-driven methods, so it is difficult to model the semantics of sparse entities well due to the data scarce. While in our model, by splitting entities into fine-grained components, each component of spare entities could appear much more frequently; 2) Besides that, in order to alleviate the sparse problem, our model predicts entities not only based on the given entity-relation pair (such as (`melbourne institute of technology`, `location`)), but also incorporates the corresponding analogy-triples (such as (`melbourne airport`, `location`, `mel-`

`bourne`)). Specifically, first the entities are split into fine granularities which are helpful to transfer semantics among entities. Then analogy-triples are extracted from KGs to enhance the modeling of entities. Furthermore, the knowledge graph completion is conducted by the sequence-to-sequence Transformer model and finally the predicted entities are directly generated.

The contribution of this paper can be summarized in the following:

- In order to alleviate the sparse problem, we propose a fine-grained knowledge graph completion method for sparse entities. The entities are split into fine granularities, which are helpful for semantic transfer among entities.
- The knowledge graph completion is completed by the sequence-to-sequence Transformer model. In the model, analogy-triples are extracted from KGs and are incorporated to enhance the modeling of entities.

2 Transformer Model

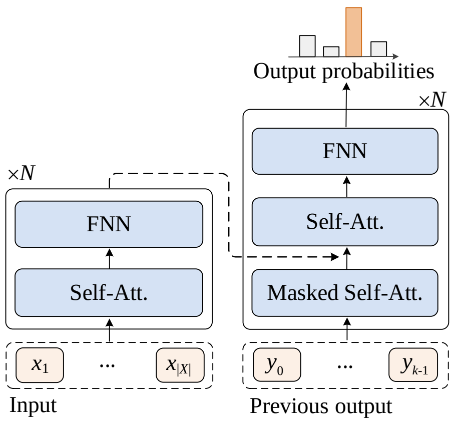

Fig. 2. The framework of Transformer model.

Transformer [21] is the state-of-the-art sequence-to-sequence model and achieves excellent performance in multiple fields due to its self-attention mechanism [24, 32]. As illustrated in Fig.2, the structure is composed of an encoder and a decoder.

Encoder First, the input sequence $X = \{x_1, \cdots, x_{|X|}\}$ is first initialized to embeddings $\mathbf{X} = [\mathbf{x}_1; \cdots; \mathbf{x}_{|X|}]$ ($\mathbf{X} \in \mathbb{R}^{d \times |X|}$)[1]. Then $\mathbf{X}$ is feed into the encoder.

[1] In this paper, the bold characters represent the embeddings in the model, and d is the dimension of embeddings.

The encoder is composed of N identical layers, and each layer is composed of two main sub-layers: self-attention sub-layer, and feed-forward sub-layer.

In the self-attention sub-layer, the input $\mathbf{X}$ is transformed to three matrices: query $\boldsymbol{Q}$, key $\boldsymbol{K}$, and value $\boldsymbol{V}$. Then they are encoded by the attention function, and the hidden states $\boldsymbol{M}_X$ is obtained. Formally,

$$\boldsymbol{M}_X = Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{g_k}}\boldsymbol{V}) \tag{1}$$

where $\frac{1}{\sqrt{g_k}}$ is the scaling factor.

In the feed-forward sub-layer, the hidden states $\boldsymbol{M}_X$ is processed by linear transformations and ReLU activation:

$$\boldsymbol{F}_X = FNN(\boldsymbol{M}_X) = max(0, \boldsymbol{M}_X\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2 \tag{2}$$

where $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, $\boldsymbol{b}_1$, and $\boldsymbol{b}_2$ are trainable parameters.

Decoder The decoder is also composed of N identical layers, and each layer mainly contains three sub-layers. The first sub-layer is the masked self-attention sub-layer. In this sub-layer, the sequence generated by the decoder in previous steps (i.e., $\{y_0, \cdots, y_{k-1}\}$, where y_0 is a special token at the beginning) are processed by Eq. (1). The output hidden states of this sub-layer is denoted by $\boldsymbol{M}_Y$. Then in the self-attention sub-layer, the hidden states obtained by encoder $\boldsymbol{F}_X$ is integrated with the hidden states of the former sub-layer $\boldsymbol{M}_Y$. This procedure can be presented by:

$$\boldsymbol{F}_Y = Attention(\boldsymbol{M}_Y, \boldsymbol{F}_X, \boldsymbol{F}_X) \tag{3}$$

Then $\boldsymbol{F}_Y$ is processed in the feed-forward sub-layer as Eq. (2). Finally, the decoder predicts the output probability of a token by:

$$P(y_k|X, Y_{<k}) = softmax(\boldsymbol{W}_3\boldsymbol{F}_Y + \boldsymbol{b}_3) \tag{4}$$

where $\boldsymbol{W}_3$ and $\boldsymbol{b}_3$ are trainable parameters. y_k denotes the output token in the k-th decode time, and $Y_{<k}$ is the output sequence in previous steps.

The procedure of predicting a sequence $Y = \{y_1, \cdots, y_{|Y|}\}$ can be represented by:

$$P(Y|X;\theta) = \prod_{k=1}^{K} P(y_k|Y_{<k}, X;\theta) \tag{5}$$

Note that all these sub-layers above are followed by residual connection and layer normalization, which are omitted for simplicity. More details can be seen in [21].

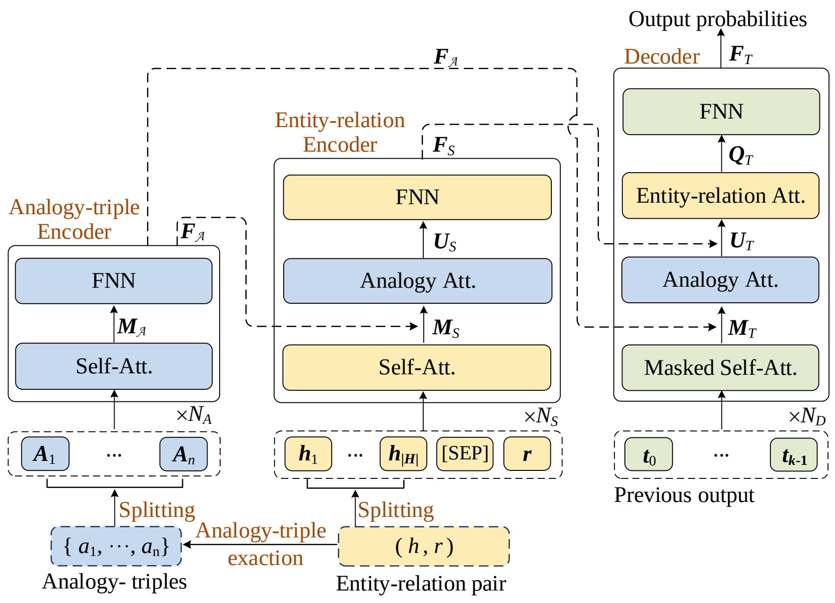

Fig. 3. The framework of our model. Given an entity-relation pair (h, r), first the entity h is split into fine granularities. Then corresponding analogy-triples are extracted from KG. Afterwards, the analogy-triples and entity-relation pair are encoded respectively, and then are integrated in the entity-relation encoder. Finally, in the decoder, previous output, hidden states of analogy-triples and entity-relation pair are all incorporated to finally generate the predicted entities.

3 Methodology

3.1 Problem Formulation

Given a KG $\mathcal{K} = \{\mathcal{E}, \mathcal{R}\}$, where $\mathcal{E}$ represents the set of entities, and $\mathcal{R}$ denotes the set of relations. KG is represented by a lot of triples (h, r, t), where entities h, $t \in \mathcal{E}$, and relation $r \in \mathcal{R}$. In this study we mainly focus on the link prediction task. Given an entity-relation pair (h, r), link prediction aims to predict the missing tail entity t.

3.2 Proposed Method

The proposed model is composed of five main modules: fine-grained splitting, analogy-triple extraction, analogy-triple encoder, entity-relation encoder, and decoder. First, all the entities are split into fine granularities. Then analogy-triples are extracted from KG based on the entity-relation pair. Afterwards, the sequences of entity-relation pair and analogy-triples are encoded and integrated. Finally, the sequences of predicted entities are generated by the decoder. The framework is illustrated in Fig. 3.

Fine-grained Splitting Fine-grained splitting is to split entities into fine granularities. In our model, we implement three different levels of granularity as follows:

- **Sub-entity granularity**. Under this granularity, each unit may contain multiple words. Take the entity `teahouse in chinatown` as an example, it can be split to {`teahouse in`, `chinatown`}.
- **Word granularity**. Word is the basic unit, and each entity is split by the words. Consider the entity `teahouse in chinatown` again, it is split to {`teahouse`, `in`, `chinatown`}.
- **Sub-word granularity**. Under this granularity, a word can be split into multiple sub-words. For example, the entity above can be split to {`tea`, `house`, `in`, `china`, `town`}.

To achieve fine-grained splitting, we introduce the text compression algorithm *byte pair encoding* (BPE) [15] in our model. In this algorithm, all the words are first split into characters, then two characters with the highest connection frequency are merged based on the statistics. Merging is conducted iteratively until the number of steps is reached or there is no more combination[2]. In this module, the BPE algorithm is modified to make it applicable for different levels of granularity on entities.

In this way, fine-grained splitting is applied on all the entities of KG. The entity h can be transformed into the sequence H, formally:

$$\begin{aligned} H &= Splitting(h), \\ H &= \{h_1, \cdots, h_{|H|}\} \end{aligned} \tag{6}$$

Thus the entity-relation pair (h, r) is transformed to sequence $\{H, r\}$. Similarly, for a triple a, the head and tail entities of a are also split into sequences, and we denote the corresponding sequence of triple a by A.

Analogy-triple Extraction This module is to extract a set of analogy-triples for each entity-relation pair (h, r) based on the corresponding sequence $\{H, r\}$. Specifically, the head entities of analogy-triples are similar with H, and the relations contained in analogy-triples is same as r. For example, for the given entity-relation pair `(melbourne institute of technology, location)`, the corresponding analogy-triple set may be `{(melbourne airport, location, melbourne), (melbourne park, location, melbourne)}`. The analogy-triples extraction can be formally represented by:

$$\mathcal{A} = \{A | Relation(A) = r, sim(Head(A), H) > \gamma\} \tag{7}$$

where $\mathcal{A}$ is a set which contains multiple sequences of analogy-triples, and A $(A \in \mathcal{A})$ is the sequence of an analogy-triple. $Head(A)$ and $Relation(A)$ are

[2] Similarly, for the sub-entity granularity, entities are first split into words, and then sub-entities can be obtained through the combination of words.

sequences of head entity and relation of A respectively. $sim(\cdot,\cdot)$ is the similarity function of entities, and the Levenshtein distance based similarity is used in this paper. γ is the similarity threshold. The maximum number of analogy-triples is set to n. In this module, a set of analogy-triples $\mathbb{A} = \{a_1, \cdots, a_n\}$ (a_i is the triple of KG $\mathcal{K}$) are extracted, and the corresponding sequences with the same level of granularity are denoted by $\mathcal{A} = \{A_1, \cdots, A_n\}$.

Analogy-triple Encoder The goal of the analogy encoder is to obtain the hidden states of analogy-triples and extract their features. Given the sequence of analogy-triples $\mathcal{A} = \{A_1, \cdots, A_n\}$, they are transformed to embeddings $\boldsymbol{\mathcal{A}} = [\boldsymbol{A}_1; \cdots; \boldsymbol{A}_n]$ ($\boldsymbol{\mathcal{A}} \in \mathbb{R}^{d\times|\mathcal{A}|}$). Then $\boldsymbol{\mathcal{A}}$ is feed into the analogy-triple encoder. The analogy-triple encoder is composed of N_A identical layers, and each layer contains the self-attention sub-layer and feed-forward sub-layer, which are same as Eq. (1) and Eq. (2). We denote the output hidden states by $\boldsymbol{F}_{\mathcal{A}}$.

Entity-relation Encoder The entity-relation encoder first encodes the sequence of entity-relation pair, and then integrates the hidden states of analogy-triples with that of entity-relation pair. The structure of the entity-relation encoder is N_S identical layers, and each layer includes three sub-layers: self-attention sub-layer, analogy attention sub-layer, and feed-forward sub-layer. First, the sequence of entity-relation pair $\{H, r\}$ is separated by a special token `[SEP]`, and the sequence is transformed to $S = \{H, \texttt{[SEP]}, r\}$. Then they are initialized to embeddings $\boldsymbol{S}(\boldsymbol{S} \in \mathbb{R}^{d\times|S|})$.

In the self-attention sub-layer, $\boldsymbol{S}$ is encoded and transformed to the matrix $\boldsymbol{M}_S$ ($\boldsymbol{M}_S \in \mathbb{R}^{d\times|S|}$) according to Eq. (1).

The difference between the entity-relation encoder and the analogy-triple encoder is that there is an extra analogy attention sub-layer. In this sub-layer, the hidden states of analogy-triples $\boldsymbol{F}_{\mathcal{A}}$ is attended by the hidden states $\boldsymbol{M}_S$. Formally:

$$\boldsymbol{U}_S = \mathit{Attention}(\boldsymbol{M}_S, \boldsymbol{F}_{\mathcal{A}}, \boldsymbol{F}_{\mathcal{A}}), \boldsymbol{U}_S \in \mathbb{R}^{d\times|S|} \tag{8}$$

Then the hidden states $\boldsymbol{U}_S$ is processed by the feed-forward sub-layer according to Eq. (2). Finally, the output of entity-relation encoder is denoted by $\boldsymbol{F}_S$ ($\boldsymbol{F}_S \in \mathbb{R}^{d\times|S|}$).

Decoder The decoder aims to predict the missing entities for each entity-relation pair. The input of decoder includes three parts: the output sequences in previous decoding steps (a special beginning hidden states $\boldsymbol{t}_0$ ($\boldsymbol{t}_0 \in \mathbb{R}^{d\times 1}$) is used in the first step), the hidden states of analogy-triples $\boldsymbol{F}_{\mathcal{A}}$ and entity-relation pair $\boldsymbol{F}_S$. The decoder is composed of N_D layers, and each layer includes four main sub-layers: self-attention sub-layer, analogy attention sub-layer, entity-relation attention sub-layer, and feed-forward sub-layer. Finally the outputs of decoder are the predicted sequences of entities.

First, the sequence predicted in previous $k-1$ steps $\boldsymbol{T}_{<k} = \{t_0, t_1, \cdots, t_{k-1}\}$ is processed by the self-attention layer according to Eq. (1), and the output of this sub-layer is denoted by $\boldsymbol{M}_T$ ($\boldsymbol{M}_T \in \mathbb{R}^{d\times k}$).

Then in the analogy attention layer, the hidden states $\boldsymbol{M}_T$ is used to attend the hidden states of analogy-triples:

$$\boldsymbol{U}_T = Attention(\boldsymbol{M}_T, \boldsymbol{F}_{\mathcal{A}}, \boldsymbol{F}_{\mathcal{A}}), \boldsymbol{U}_T \in \mathbb{R}^{d \times k} \tag{9}$$

Next the hidden states $\boldsymbol{U}_T$ is integrated with the entity-relation hidden states $\boldsymbol{F}_S$ in the entity-relation attention layer:

$$\boldsymbol{Q}_T = Attention(\boldsymbol{U}_T, \boldsymbol{F}_S, \boldsymbol{F}_S), \boldsymbol{Q}_T \in \mathbb{R}^{d \times k} \tag{10}$$

Finally, the hidden states is processed in the feed-forward sub-layer and the final output of decoder is represented by $\boldsymbol{F}_T$.

The probability distribution at the k-th decode step is predicted by:

$$P(\boldsymbol{t}_k | A, (h, r), T_{<k}; \theta) = softmax(\boldsymbol{W}\boldsymbol{F}_T + \boldsymbol{b}) \tag{11}$$

where $\boldsymbol{W}$ and $\boldsymbol{b}$ are trainable parameters. Thus the sequence of predicted entity can be obtained by $T = \{t_1, \cdots, t_{|T|}\}$. Similar with [14], the output sequences of decoder are always exactly the entities, so the generated sequences are linked to entities through exactly string matching.

3.3 Training

During training, the parameters are optimized by minimizing the negative log-likelihood objective function:

$$NLL(\theta) = -\frac{1}{|T|} \sum_{k=1}^{|T|} logp(t_k | T_{<k}, \mathbb{A}, (h, r); \theta) \tag{12}$$

where $\{\mathbb{A}\}$ denotes the analogy-triple sets, $\{(h, r)\}$ represents the entity-relation pairs, and $\{t\}$ is the set of corresponding tail entities. $T_{<k}$ is the decoded sequence in previous k steps. θ represents all the parameters in the model.

4 Experiment

4.1 Experiment Setup

Datasets The experiments are conducted on three public datasets: 1) FB15k-237 [19] is a sparse dataset which is drieved from FB15k through removing the inverse triples. 2) YAGO3-10-dr [1] is a sparse dataset modified from YAGO3-10. Similar with FB15k-237, some duplicate relations are removed from YAGO3-10. 3) Wikidata5M [14,25] is a large-scale benchmark dataset. The details of datasets are displayed in Table 1.

Table 1. Statistics of datasets.

	#entity	#relation	#train	#valid	#test
FB15k-237	14,541	237	272,115	17,535	20,466
YAGO3-10-dr	122,837	36	732,556	3,390	3,359
Wikidara5m	4,818,503	828	21,343,515	5,357	5,321

Baselines We compare our method with three types of baselines as follows:

- **Embedding based methods**. TransE [3] and RotatE [17] are translation-based representation learning models. DisMult [28] and ComplEX [20] are tensor decomposition based methods.
- **Rule based methods**. Neural-LP [29] transforms the procedure of logic rule learning to be differentiable, and proposes an end-to-end differentiable model for rule learning. ProbCBR [5] extracts rules with probabilities from other entities and then apply the rules to the given entity-relation pair.
- **Pre-trained language model based methods**. KGT5 [14] adopts sequence-to-sequence pre-trained model T5 [13] to achieve knowledge graph completion. SimKGC [23] introduces negative samples and textual description of entities and relations in the pre-trained language model BERT.

Evaluating Metrics We use mean reciprocal rank (MRR), and Hits@n ratio to evaluate the performance. These metrics are generally used to measure the quality of ranks. The results are evaluated under filtered settings[3].

Implementation Details Our model is developed based on the papers [18,31], and the hyper-parameters of Transformer model are same as [18]. The hidden states d is set to 512, and the batch size is 4096. The iterations of layers in two encoders (N_A and N_S) and decoder (N_D) are set to 6. In the fine-grained splitting module, for sub-word granularity, the BPE algorithm is implemented for 8000 steps on FB15k-237, and 30000 steps on YAGO3-10-dr and Wikidata5M. In the analogy-triples extraction module, the similarity threshold γ is set to 0.5, and the maximum number of analogy-triples n is 3.

4.2 Performance Comparison with Baselines

Table 2 shows the link prediction results on three datasets by different methods. Lines 1-4 are results by embedding based methods[4]. Lines 5-6 are results of rule based methods[5]. Lines 7-8 show the performance obtained by pre-trained

[3] More details can be seen in paper [3].

[4] These results are quoted from papers [1, 14].

[5] The results of these two models are obtained through our implementation using the open source codes. Note that the results on Wikidata5M is empty because it is difficult to extend to large-scale KGs for these two rule-based methods [4].

Table 2. The main results of link prediction.

#	Method	FB15k-237			YAGO3-10-dr			Wikidata5M		
		MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10	MRR	Hit@1	Hit@10
1	**TransE** [3]	0.288	0.198	0.441	0.190	0.136	0.323	0.253	0.170	0.392
2	**DisMult** [28]	0.238	0.199	0.446	0.192	0.133	0.307	0.253	0.208	0.334
3	**ComplEX** [20]	0.249	0.194	0.450	0.201	0.143	0.315	0.281	0.228	0.373
4	**RotatE** [17]	0.337	0.241	0.533	0.214	0.153	0.332	0.290	0.234	0.390
5	**Neural-LP** [29]	0.240	0.251	0.362	0.187	0.132	0.297	-	-	-
6	**ProbCBR** [5]	0.231	0.187	0.320	0.181	0.128	0.284	-	-	-
7	**SimKG** [23]	0.336	0.249	0.511	-	-	-	0.358	0.313	0.441
8	**KGT5** [14]	0.276	0.210	0.414	0.211	0.151	0.327	0.300	0.267	0.365
9	**FineKGC-Base**	0.379	0.289	0.556	0.254	0.186	0.387	0.369	0.306	0.443
10	**FineKGC-Ana.**	**0.389**	**0.299**	**0.567**	**0.273**	**0.204**	**0.404**	**0.387**	**0.321**	**0.452**

language model based methods[6]. In Line 9, our model is implemented without incorporating analogy-triples. In Line 10, the analogy-triples for each entity-relation pair are extracted and attended in the model. In our models, the entities are split into the sub-word granularity. The best performance is shown in bold fonts. We can reach the following conclusions:

1) Considering the overall results, our model `FineKGC-Ana.` (Line 10) achieves the best performance on three datasets, suggesting the superiority of the proposed model. Specifically, compared with `RotatE` (`RotatE` performs the best on 5 out of 9 metrics in baselines), our proposed model achieves the maximum improvement of 0.072 (Hits@10) on YAGO3-10-dr and 0.087 (Hits@1) on Wikidata5M. The results demonstrate the effectiveness of our model.

2) Compare the performance of our models in Lines 9-10, the performance in line 10 are better than that of line 9 on all three datasets. The maximum improvement is 0.019 on metric MRR of YAGO3-10-dr. The performance shows that it is helpful to incorporate analogy-triples in our model.

4.3 Performance with Different Frequencies

Our model mainly aims to improve the performance of knowledge graph completion on sparse entities. To further verify the effect of our model on sparse entities, in this experiment, we compare the performance of models on entities

[6] For SimKGC, the results on FB15k-237 and Wikidata5M is obtained from its original paper [23]. For KGT5, the results on FB15k-237 and Wikidata5M are quoted from [14].

with different frequencies. Specifically, first the validation set of FB15k-237 is divided into four subsets according to the frequency of entities (the frequency of entities ranges in 1-10, 10-20, 20-30, and greater than 30, respectively). Then we randomly sample 500 test data from each subset. The results of different methods on these four test sets are plotted in Fig. 4.

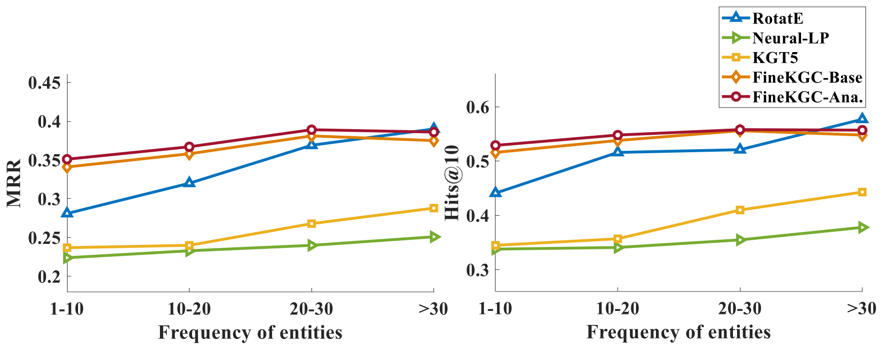

Fig. 4. The performance of models on entities with different frequencies.

In Fig. 4, the X-axises denote the frequencies of entities, and the Y-axises depict the MRR (left) and Hits@10 (right) respectively. We can see when the frequency of entities is low (when the frequency ranges in 1-10), the MRR and Hits@10 of three baselines are small, which demonstrates that the sparse problem is a challenge for knowledge graph completion. While our method can obtain remarkable progress than baselines on these sparse entities.

4.4 Performance with Different Levels of Granularity

As introduced in Section 3.2, the entities are split into different granularities. To test the corresponding performance, we conduct experiments on the validation set of YAGO3-10-dr with different levels of granularity. Table 3 shows the performance.

From the results, we achieve the following conclusions:

1) The comparison under different levels of granularity shows that the model `FineKGC-Ana.` always performs better than `FineKGC-Base`. The results indicate that incorporating analogy-triples in the model is effective under different levels of granularity.

2) Furthermore, the sub-word granularity beats the other two levels of granularity by both `FineKGC-Base` and `FineKGC-Ana.`. The reason is supposed that the sub-word granularity is finer than word and sub-entity granularities, which is more helpful for semantic transfer among entities.

Table 3. Performance with different levels of granularity.

Gran.	Model	MRR	Hits@1	Hits@10
sub-word	FineKGC-Base	0.254	0.186	0.387
	FineKGC-Ana.	0.273	0.204	0.404
word	FineKGC-Base	0.226	0.167	0.330
	FineKGC-Ana.	0.238	0.178	0.350
sub-entity	FineKGC-Base	0.210	0.159	0.313
	FineKGC-Ana.	0.217	0.165	0.326

4.5 Performance with Different Number of Analogy-triples

In this experiment, we discuss the influence of the analogy-triple number on the performance of our model. Fig. 5 plots the MRR (subfigure (a)), Hits@1 (subfigure (b)) and Hits@10 (subfigure (c)) of our model when the number of analogy-triples is 0, 1, 3, and 6 respectively.

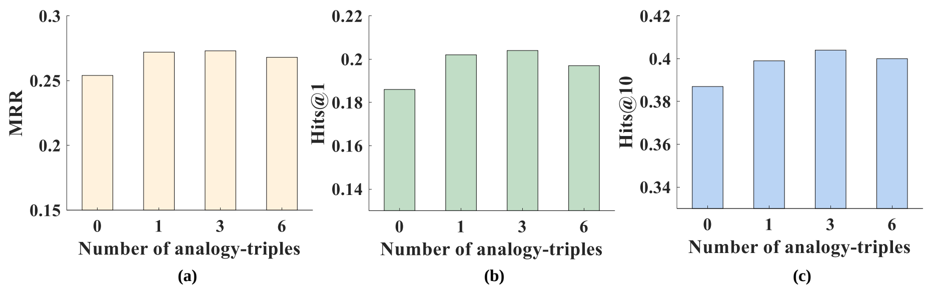

Fig. 5. The performance of our model with different numbers of analogy-triples.

From Fig. 5, we can see the results of model incorporating analogy-triples (the numbers of analogy-triples are 1,3, and 6) are greater than that of model without analogy-triples (the number of analogy-triples is 0). These results show that it is effective to introduce analogy-triples in our model. Moreover, our proposed model achieves the best performance when selecting 3 most similar analogy-triples.

4.6 Case Study

We choose three examples which are predicted correctly by our model on sparse entities (the frequencies of entities in these entity-relation pairs are all smaller than 10). The details are displayed in Table 4.

Table 4. Case study of our model on sparse entities.

Case 1	
Entity-relation pair	(shakti kapoor, nationality)
Predicted entity	india
Analogy-triples	•(shammi kapoor, nationality, india) •(shashi kapoor, nationality, india) •(shobha kapoor, nationality, india)
Case 2	
Entity-relation pair	(chaozhou, location)
Predicted entity	china
Analogy-triples	•(shuozhou, is located in, china) •(chuzhou, is located in, anhui) •(changzhou, is located in, jiangsu)
Case 3	
Entity-relation pair	(new york university school of law, location citytown)
Predicted entity	new york city
Analogy-triples	•(johns hopkins university school of medicine, location citytown, baltimore) •(stanford university school of humanities and sciences, location citytown, stanford) •(boston university school of law, location citytown,boston)

In Case 1, our model correctly predicts thes `nationality` of the entity `shakti kapoor`. As `kapoor` is the common seen surname of India, and in the analogy-triples, all the head entities which contain the fine-grained component `kapoor` has Indian nationality. The semantic of `kapoor` in analogy-triples can be transferred to the sparse entity `shakti kapoor`. As a consequence, the `nationality` can be correctly predicted.

In Case 2, all the entities in the entity-relation pair and analogy-triples contain the fine-grained component `zhou` and the tail entities are all in China. As a consequence, the semantic of the fine-grained component `zhou` can be transferred, and the `location` of `chaozhou` may be China.

The analogy-triples in Case 3 indicate that the location information is usually implied in the fine-grained components of university schools. Thus the location of entity `new york university school of law` is predicted to be `new york city`.

5 Related Work

Embedding based methods Embedding based methods aims at learning low-dimensional embeddings for entities and relations in KGs, and then predicting the missing triples through scoring candidate triples based on the learnt embeddings. TransE [3] is one of the representative methods which models the relations as translation operations from the head entities to tail entities. Then TransH [26], TransD [7] and TranSparse [8] extend TransE from different perspectives. DisMult [28] models the procedure of scoring triples as tensor factorization, and designs a bilinear formulation to score triples based on semantic matching. ComplEX [20] extends DisMult through mapping entities and relations to complex-valued space to model symmetric and anti-symmetric relations. RotatE [17] models relations as rotation operations from head to tail entities in the complex-valued space.

Besides that, several methods improve the representation learning through introducing extra resources, such as textual descriptions [10,16]. However, these methods utilize entity as the basic granularity, which face the semantic under-transfer problem. In our model, entities are split into different levels of granularity, thus the semantics of the fine-grained components in sparse entities can be transferred from other similar entities.

Rule based based methods Other stream of models aim to learn logic rules from KGs, and knowledge graph completion is conducted through applying the rules on existing data. AMIE [11] first extracts multiple candidate rules and then designs multiple statistic-based metrics (pca-confidence and head coverage) to evaluate the quality of candidate rules. Neural-LP [29] proposes an end-to-end differentiable model to learn logic rules. Rudik [12] extends the form of rules, and the obtained negative rules can be used for error recognition. ProbCBR [5] is a case-based method, the rules are obtained from entities in the same cluster, and the quality of rules are also obtained based on the statistic of the corresponding cluster. However, these models learn logic rules based on the frequency patterns of KG. For sparse entities, it is difficult to obtain qualified logic rules due to data scarcity.

Pre-trained language model based methods Pre-trained language models (PLMs) have recently attracted enormous interest due to its large-scale background knowledge. KG-BERT [30] employs BERT [9] to complete knowledge graphs. First BERT is fine-tuned by knowledge graphs for relation prediction task and link prediction task respectively, and then the entity and relation is used as input (separated by token `[SEP]`), then the output embedding of special token `[CLS]` is used to predicate the other entity. StAR [22] also utilizes BERT to learn contextual embeddings of entities. Besides that, a scoring module is designed to learn both contextualized and structured knowledge of KGs. KGT5 [14] introduces the T5 [13] model in knowledge graph completion task and question answering task simultaneously. For KGC task, the input is the verbalized head/tail entities and the relations, then the output is the tail/head entities. SimKGC [23] incorporates contrastive learning on pre-trained language model BERT to implement knowledge graph completion. Besides that, textual descriptions of head entities and relations are also utilized to model the relationaware semantics. Different from these methods, there is no need for our model to utilize the pre-trained language models and large amount of textual resources. The experimental results also demonstrate the superiority of our model. In the future we will explore to incorporate pre-trained models with our method.

6 Conclusion

In this paper, we propose an analogy-triple enhanced fine-grained Transformer model for sparse knowledge graph completion. First, entities are split with different levels of granularity. In this way, the semantics of sparse entities can be modeled and transferred from other similar entities. Then we introduce the analogy-triples to enhance the modeling of entities. Finally, knowledge graph completion task is conducted by the sequence-to-sequence model. In the model, the fine grained entity-relation pairs and analogy-triples are jointly attended to generate predicted tail entities. The experimental results demonstrate that our model outperforms existing methods on sparse knowledge graph completion.

Acknowledgements This work was supported by NSFC under grant 61932001 and U20A20174. Lei Zou is the corresponding author of this paper.

References

1. Akrami, F., Saeef, M.S., Zhang, Q., Hu, W., Li, C.: Realistic re-evaluation of knowledge graph completion methods: An experimental study. In: SIGMOD. pp. 1995–2010 (2020)
2. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: SIGMOD. pp. 1247–1250 (2008)
3. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. NeurIPS **26** (2013)
4. Chen, X., Jia, S., Xiang, Y.: A review: Knowledge reasoning over knowledge graph. Expert Syst. Appl. **141**, 112948 (2020)
5. Das, R., Godbole, A., Monath, N., Zaheer, M., McCallum, A.: Probabilistic case-based reasoning for open-world knowledge graph completion. In: EMNLP. pp. 4752–4765 (2020)
6. Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., Zhang, W.: Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In: SIGKDD. pp. 601–610 (2014)
7. Ji, G., He, S., Xu, L., Liu, K., Zhao, J.: Knowledge graph embedding via dynamic mapping matrix. In: ACL. pp. 687–696 (2015)
8. Ji, G., Liu, K., He, S., Zhao, J.: Knowledge graph completion with adaptive sparse transfer matrix. In: AAAI (2016)
9. Kenton, J.D.M.W.C., Toutanova, L.K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT. pp. 4171–4186 (2019)
10. Kong, F., Zhang, R., Guo, H., Mensah, S., Hu, Z., Mao, Y.: A neural bag-of-words modelling framework for link prediction in knowledge bases with sparse connectivity. In: WWW. pp. 2929–2935 (2019)

11. Lajus, J., Galárraga, L., Suchanek, F.: Fast and exact rule mining with amie 3. In: ESWC. pp. 36–52. Springer (2020)
12. Ortona, S., Meduri, V.V., Papotti, P.: Rudik: Rule discovery in knowledge bases. VLDB Endowment **11**(12), 1946–1949 (2018)
13. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J., et al.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**(140), 1–67 (2020)
14. Saxena, A., Kochsiek, A., Gemulla, R.: Sequence-to-sequence knowledge graph completion and question answering. In: ACL. pp. 2814–2828 (2022)
15. Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. In: ACL. pp. 1715–1725 (2016)
16. Shi, B., Weninger, T.: Open-world knowledge graph completion. In: AAAI. vol. 32 (2018)
17. Sun, Z., Deng, Z.H., Nie, J.Y., Tang, J.: Rotate: Knowledge graph embedding by relational rotation in complex space. In: ICLR (2018)
18. Tan, Z., Zhang, J., Huang, X., Chen, G., Wang, S., Sun, M., Luan, H., Liu, Y.: Thumt: an open-source toolkit for neural machine translation. In: AMTA. pp. 116–122 (2020)
19. Toutanova, K., Chen, D.: Observed versus latent features for knowledge base and text inference. In: CVSC. pp. 57–66 (2015)
20. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: ICML. pp. 2071–2080. PMLR (2016)
21. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. NeurIPS **30** (2017)
22. Wang, B., Shen, T., Long, G., Zhou, T., Wang, Y., Chang, Y.: Structure-augmented text representation learning for efficient knowledge graph completion. In: WWW. pp. 1737–1748 (2021)
23. Wang, L., Zhao, W., Wei, Z., Liu, J.: Simkgc: Simple contrastive knowledge graph completion with pre-trained language models. In: ACL. pp. 4281–4294 (2022)
24. Wang, S., Dang, D.: A generative answer aggregation model for sentence-level crowdsourcing task. IEEE Trans. Knowl. Data Eng. (2022)
25. Wang, X., Gao, T., Zhu, Z., Zhang, Z., Liu, Z., Li, J., Tang, J.: Kepler: A unified model for knowledge embedding and pre-trained language representation. TACL **9**, 176–194 (2021)
26. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: AAAI. vol. 28 (2014)
27. Xue, B., Zou, L.: Knowledge graph quality management: a comprehensive survey. IEEE Trans. Knowl. Data Eng. (2022)
28. Yang, B., Yih, S.W.t., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. In: ICLR (2015)
29. Yang, F., Yang, Z., Cohen, W.W.: Differentiable learning of logical rules for knowledge base reasoning. NeurIPS **30** (2017)
30. Yao, L., Mao, C., Luo, Y.: Kg-bert: Bert for knowledge graph completion. arXiv preprint arXiv:1909.03193 (2019)
31. Zhang, J., Luan, H., Sun, M., Zhai, F., Xu, J., Zhang, M., Liu, Y.: Improving the transformer translation model with document-level context. In: EMNLP. pp. 533–542 (2018)
32. Zhao, Y., Zhang, J., Zhou, Y., Zong, C.: Knowledge graphs enhanced neural machine translation. In: IJCAI. pp. 4039–4045 (2021)