Adaptive Attentional Network for Few-Shot Relational Learning of Knowledge Graphs

Ruixin Ma 1,2, Zeyang Li 1,2, Yunlong Ma 1,2, Hao Wu 1,2, Mengfei Yu 1,2 and Liang Zhao 1,2,4

1 School of Software Technology, Dalian University of Technology, Dalian 116024, China; maruixin@dlut.edu.cn (R.M.); lizeyang@mail.dlut.edu.cn (Z.L.); m1804069847@mail.dlut.edu.cn (Y.M.); WuHao15904655726@mail.dlut.edu.cn (H.W.); yumengfei@mail.dlut.edu.cn (M.Y.)
2 The Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian 116600, China
3 Correspondence: liangzhao@dlut.edu.cn

Abstract: Few-shot knowledge graph reasoning is a research focus in the field of knowledge graph reasoning. At present, in order to expand the application scope of knowledge graphs, a large number of researchers are devoted to the study of the multi-shot knowledge graph model. However, as far as we know, the knowledge graph contains a large number of missing relations and entities, and there are not many reference examples at the time of training. In this paper, our goal is to be able to infer the correct entity given a few training instances, or even only one training instance is available. Therefore, we propose an adaptive attentional network for few-shot relational learning of knowledge graphs, extracting knowledge based on traditional embedding methods, using the Transformer mechanism and hierarchical attention mechanism to obtain hidden attributes of entities, and then using a noise checker to filter out unreasonable candidate entities. Our model produces large performance improvements on the NELL-One dataset.

Keywords: few-shot; one-shot; knowledge graph reasoning; Transformer

1. Introduction

A knowledge graph (KG) is composed of a large number of triples in the form (head entity, relation, tail entity) (h, r, t), and this structured knowledge is crucial for many downstream applications such as question answering and the semantic web. In recent years, the proposed knowledge graphs all represent each piece of information as a triple relation, such as DBpedia [1], Free-base [2], NELL [3], Probase [4], Wikidata [5], etc. They are obtained by mining through knowledge measurement, data mining, information processing, and graph rendering to provide valuable references for display or application. However, in real KGs, they are highly incomplete [6], a large part of the relations in KGs are long-tailed relations, and there are few training samples [7].

Some excellent methods have been proposed in the field of few-shot knowledge graph reasoning. They mainly include metric-based algorithms and meta-learning algorithms.

For example, GMatching [7] is the first metric-based method, which targets a query triple (h, r, t) that requires a reference entity pair in the task relation r to predict the true tail entity. The model assumes that adjacent entities of the source entity are equally important. This method is the first to achieve knowledge graph reasoning on a single sample and achieves suitable results on public datasets. HAF [8] is an effective hybrid attention mechanism learning framework that acquires highly interpretable entity aggregation information through an attention module. Among them, the entity enhancement encoder learns the hidden features of entities in different scenarios, adjacent encoders update en-
tity features to accommodate different training processes, and then use the matching processor to infer entities. However, this model ignores that the same entity has different weights in different relationships or different tasks.

Among meta-learning-based methods, MetaR [9] uses a meta-learning encoder for inference, using gradient elements to change relation elements to obtain the minimum value of the loss function, thereby speeding up the learning process of the model. The goal of this model is to map small-sample entities to relational elements better and then use gradient elements to speed up the convergence of relational elements. However, both works ignore the different contributions of neighbors to entities. However, both works ignore the different contributions of neighbors to entities. FSRL [10] learns entity embeddings through domain graph structure and fixed weights to infer real entities, given a small set of reference entity pairs for each relation. However, this method does not work very well for one-shot cases. Since neighbors are assigned fixed weights, they cannot accommodate different inference relations. This paper believes that the accuracy of inference results is not only affected by the dominant features of the entity itself but also affected by the recessive features. At the same time, one-shot task scenarios are more common in the real world than few-shot tasks. For example, the features of an entity’s neighboring neighbors can cascade to affect the overall features of the entity [11,12].

Figure 1 is a schematic diagram of the attribute relationship of the knowledge graph of Ma Yun as the source node. The triple relationship on the left of Figure 1 is “Work_with”, “Proxy_for”, and “Partner”, all of which belong to Ma Yun’s business relationship. The triple relationship on the right side of Figure 1 is “Marry_to”, “Has_Child”, and “Is_child_of”, all of which belong to Ma Yun’s family relationship. From Figure 1, it can be seen that the two types of relations are very different, and the relations of the same attribute are closer in the embedding vector space. If the model query task is (Ma Yun, Ceo_of,?) triplet, then model training should pay more attention to Ma Yun’s business relationship. Therefore, the tail entity of the business relationship should have a higher weight than the tail entity of the family relationship.

![Figure 1. Schematic diagram of knowledge graph reasoning attribute relationship.](image)

The same attribute relation triples should also have different weights for the source entity. In Figure 1, the triples (Ma Yun, Work_With, Sun Tongyu), (Ma Yun, Proxy_For, Alibaba), and (Ma Yun, partner, Xie Shihuang) that all belong to Ma Yun’s business relationship are in the model training process. Since Sun Tongyu and Xie Shihuang are closer to the inference result, so (Ma Yun, Proxy_For, Alibaba) should have the highest weight.

In addition to the above two weighting scenarios, it is also necessary to consider the influence of entities with different hops on the same path on the source entity. Because the entities and relations of the knowledge graph are represented by embedding vectors,
the embedding relation of triples satisfies \( h + r \approx t \). So when in triples (Ma Yun, Proxy_For, Alibaba) and (Alibaba, year of creation, 1999), the Alibaba embedding vector is closer to the embedding vector of the source entity Jack Ma than the 1999 embedding vector.

The previous methods cannot solve the above problems. The feature influence of entity neighbors on the source entity in dynamic scenarios cannot be distinguished, and the feature influence of the same attribute relationship is not considered. This paper believes that the weights of entities for different task scenarios should also be different, and at the same time, the impact of the same attribute relationship triplet on the weight of the source entity should also be greater than that of entities with different attribute relationship three groups.

In order to solve the above problems, this paper proposes an adaptive attentional network for few-shot relational learning of knowledge graphs (AAF), whose purpose is to learn a matching function that accurately infers real entities. Specifically, firstly, the Transformer mechanism [13] and the hierarchical attention mechanism are mixed to obtain the high-level domain aggregation information of entities. This module can capture the feature effects of adjacent entities in different task scenarios and different levels on the source entity. Next, to improve the source entity information, the neighbor encoder synthesizes the deep and surface features of the source entity through the domain structure. Then, a noise detection module is used to filter out unreasonable candidate entities. Finally, the matching processor first builds the negative triples, and the trained parameters are then optimized through a loss function to further infer the true entity pairs for the new relationship.

Our contributions are as follows:

1. We propose an adaptive attention mechanism to process different task scenarios by researching the features of triples in different task scenarios and at different levels in few-shot scenarios. Moreover, in order to obtain more entity feature information, we enhance the multi-level aggregation mechanism of entity surface features and deep features in the knowledge graph. Therefore, we can improve the accuracy of inference through more accurate entity feature information.

2. We design an adaptive attentional network for the few-shot relational learning model of knowledge graphs (AAF), using Transformer and hierarchical attention to learn the latent features of entities in different scenarios. The neighbor encoder updates the source entity feature information to adapt it to the reasoning process of different task scenarios. Noise check to filter unreasonable entities in the candidate set. Finally, the matching processor infers the ground truth by the candidate entity similarity scores.

3. We conduct extensive experiments to evaluate our model on the NELL-One dataset. The experimental results show that our model outperforms the baseline model in both one-shot and few-shot task scenarios, proving the effectiveness of the entity-enhanced encoder and noise checker.

2. Related Work

In this section, we briefly introduce three areas related to this paper, as follows:

2.1. Embedding Models for Relational Learning

Embedding models for relation learning can be mainly divided into two categories: translation-based models [14–18] and convolutional network-based models [19–23]. Most of the former methods first model the vector space of the knowledge graph and then infer the actual entities or relationships in the graph. RESCAL [14], DistMult [15], ComplEx [16], and RotatE [17] all want to mine the latent semantic features of triples and then model the relationship to infer the missing parts. TransE [18] is the first transfer-based representation model, and it is also an embedding-based base model, setting off a research boom in the Trans series. DistMult, ComplEx, and RotatE are proposed to further model the vector features of entities and relationships along the lines of TransE. In recent years, with
the richness of neural network shapes, more and more models based on convolutional networks have been proposed. In order to solve the problem that the shallow link prediction model lacks the ability to extract deep features, blindly increasing the embedding size will lead to overfitting. The ConvE [19] model first converts entity and relation vector representations to 2-dimensional and cat-operates and performs dropout operation on the encoded data, then performs convolution, batch normalize, etc., and then goes through a fully connected layer and uses softmax to obtain the corresponding probability. Later, ConvR [20] uses the relation as the convolution kernel to perform two-dimensional convolution on the representation of the entity, which reduces the amount of parameters of the convolutional network and enriches the interaction between entities and relations.

In order to solve the problem that ConvE does not pay attention to the global relationship of triples with the same dimension and ignores the translation characteristics of translation models, ConvKB [21] represents each triple as a three-column matrix, which is fed to the convolutional layer, and multiple The convolution kernel performs a convolution operation on it to output a feature vector, and the feature vector is multiplied by a weight vector to return the score of the triplet. Based on the idea of GCN information dissemination, R-GCN [22] is proposed as a model for knowledge graph embedding. The model uses the self-attention mechanism to solve the influence of different edge relationships on nodes, uses the convolution output of the upper layer as the convolution input of the lower layer, and designs different output layers for entity classification and link prediction tasks. To solve the problem of the inability to extract semantic features of different granularities, ConvKG [23] uses polymorphic filters to co-convolve triple embeddings to jointly learn semantic features of different granularities. Filters of different shapes cover different sizes on triple embeddings and capture different granularities of pairwise interactions between triples.

The above embedding-based models usually assume sufficient training examples for all relations and entities and do not focus on those sparse symbols. Therefore, it is not suitable for the reasoning research of few-shot knowledge graphs. Compared with these methods, our model can handle long-tail or newly added relations and can adapt to both 1-shot relational learning and few-shot relational learning at the same time.

2.2. Few-Shot Learning

Few-shot learning methods can be roughly divided into two categories: (1) metric-based methods [24–28] and (2) meta-learning-based methods [29–32]. The metric-based learning method mainly includes two modules: the embedding module and the measurement module. The samples are embedded in the vector space through the embedding module, and then the similarity score is given according to the measurement module. By calculating the distance between the to-be-classified sample and the known classified sample, find the adjacent categories to determine the classification result of the to-be-classified sample. A siamese neural network learns metrics from data and then uses the learned metrics to compare and match samples of unknown classes. The two siamese neural networks share a set of parameters and weights [24]. Matching Networks is a new type of neural network structure that combines Metric Learning and Memory Augment Neural Networks. This network uses the attention mechanism and memory mechanism to accelerate learning label prediction for unlabeled samples with few samples [25]. Based on the prototype network of mixed attention, the prototype network is used to design instance-level and feature-level attention mechanisms to highlight key instances and features, which significantly improves performance and speeds up convergence [26] DN4 (deep nearest neighbor neural network) and Different from other metric-based learning methods, DN4 finds the category closest to the input image by comparing the local descriptor between the image and the category, that is, replaces the simple image feature vector with the local description [27]. DPGN is the first GNN method to use distribution propagation, and it is a relatively novel method that uses both instance-level and distribution-level re-
lations [28]. The latter performs fast model parameter updates given the gradients of several training instances. LSTM-based meta-learning [29] learns the exact optimal algorithm through the step size of each dimension of the stochastic gradient. The main idea of MAML [30] and MT-net [31] is to train the model through two-stage training few-shot gradient update and then be able to quickly reason about new tasks.

In fact, previous few-shot learning methods are not suitable for knowledge graph inference models because some features are lost when the few-shot learning model is pooled. Then, the logical structure of the knowledge graph can improve the interpretability of the model. At the same time, this paper believes that more entity information can be obtained by mixing invisible feature vectors and explicit feature vectors. This paper uses high-order domain aggregation vectors to enhance entity-dominant features.

3. Background

In this section, the formal definition of the learning task of few-shot knowledge graph relations is introduced.

3.1. Task Formulation

Definition 1. (Knowledge Graph G) The knowledge graph \( G = (\text{Node}, \text{Edge}) \) consists of triples \((h, r, t)\) \( \subseteq E \times R \times E \). \( G \) represents the graph structure composed of all knowledge. \( \text{Node} \) represents the node in the knowledge graph, \( \text{Edge} \) represents the edge between nodes, \( E \) is the set of all entities, and \( R \) is the set of all relations.

Definition 2. (Few-shot knowledge graph relation learning task \( T \)) The task is to perform entity prediction and relation prediction through a few reference entity pairs. Entity prediction is to predict the tail entity given the head entity and relation: \((h, r, ?)\), and relation prediction is to predict the relation given the head entity and the tail entity: \((h, ?, t)\). The training task in this paper is entity prediction.

As mentioned above, the support set \( S_r = \{(h_i, t_i) \in E \times E \mid (h_i, r, t_i) \in E \times R \times E\} \) is employed in the matching processor module to infer candidate entities, and the inferred set is \( Q_r = \{r : (h_i, r, \?)\} \). In detail, while using a few training triples, we can achieve real candidate entities to rank at the top of the entity set. When inferring new candidate entities, we do not involve dataset facts that change during model training, only the currently ingested dataset.

3.2. Few-Shot Learning Settings

The purpose of this paper is to train a relational learning model that can adapt to the small-sample situation on the knowledge graph, which can reason about the real tail entities with few samples. Referring to the previous one-shot learning setting [25], we set the learning setting for this model. In this work, each training relation \( r \in R \) in the KG has its own training/testing samples: \( T_r = \{D_r^{\text{train}}, D_r^{\text{test}}\} \). We use the meta-training set \( T_{\text{meta-train}} \) to represent the task set.

The set \( D_r^{\text{train}} \) value of \( r \) includes a few sample triples \((h_1, r, t_1)\), and this paper uses this setting to evaluate the prediction of small-sample entities. In addition, each relation \( r \) corresponding to \( D_r^{\text{test}} = \{(h_i, r, t_i, C_{h_i, r}) \mid (h_i, r, t_i) \in G\} \) includes all the test triples, the correct tail entities of the query triples, and the candidate entities not detected by noise, which are all nodes existing in \( G \). Therefore, the method in this paper is validated by comparing the similarity metric scores between the query triplet \((h, r, ?)\) and the reference samples in the training set, and entities with high similarity scores are considered as inference results.

\( L_\theta(h, r, t, C_{h, r}) \) is the loss function of the correspondence, where \( \theta \) represents all the parameters that need to be trained. The training objectives of this model are:
\[
\min_{\theta} E_{T_r} \left[ \sum_{(h,r,t) \in D_{r}^{\text{test}}} L_{\theta}(h, r, t) \middle| C_{h, r}, D_{r}^{\text{train}} \right] \quad (1)
\]

where \( T_r \) represents some random samples in the training set \( T_{\text{meta-train}} \) and \( |D_{r}^{\text{test}}| \) represents the number of triples in \( D_{r}^{\text{test}} \).

According to the above assumptions, this paper extracts some relations from the training set to construct the meta-validation set of this model. Then, the background knowledge graph \( G’ \) is extracted from the complete knowledge graph \( G \), and the background graph information can be used by triples in the training process.

4. Methods

This section will describe a few-shot knowledge graph learning framework with adaptive weights and the loss function \( l \) for model training.

AAF mainly includes four parts: the first part is the entity-enhanced encoder, the second part is the neighbor encoder, the third part is the noise checker, and the fourth part is the matching processor. The most important of these are the entity-enhanced encoder and noise checker. For any query relation \( r \), the model uses the known real triples \((h_k, r, t_k)\) in the background graph and obtains the high-order domain aggregation information of the entity through the encoder of the Transformer mechanism and the hierarchical attention mechanism. For each tail entity in the reference entity pair \((h_k, t_k)\), the noise checker first ranks the candidate entities in the embedding vector space, filtering out the candidate entities that are too noisy. Then, the matching processor performs a similarity score ranking on the matching scores between the reference entity pair and the candidate entity pair to obtain the inference entity. The overall frame diagram of the model is shown in Figure 2. The upper left corner is the neighbor encoder, the dotted line is surrounded by the entity-enhanced encoder, the lower-left module is the noise checker, and the right one is the model matching processor module.

![Figure 2](image)

**Figure 2.** Overview of the AAF framework.

4.1. Entity-Enhanced Encoder
The part extracts the hidden properties of entities through local graph structure.

The existing work [32,33] has proved that constructing model modeling can improve the accuracy of actual prediction. HAF [8] has proved that a one-step neighborhood is helpful for entity prediction, but this framework does not consider the feature influence of the same relational attributes on the source entity under different training tasks. To address this issue, we design an adaptive weighted entity augmentation encoder.

In order to gain the hidden attribute representation of \( H_{r_e} \) of the triple \((h, r, t)\), this paper needs to address two weighting scenarios. The primary case is that the source entity has the same distance neighbor entities; the second case is that the source entity has the same path and different distance neighbor entities. We first use the existing embedding-based model DistMult [15] for pre-training, and then the entity and relation embedding vectors are represented as follows:

\[
\mathbf{v}_h = \text{emb}(h), \mathbf{v}_r = \text{emb}(r), \mathbf{v}_t = \text{emb}(t),
\]

where \( h, r, \) and \( t \) are the head entity, relation, and tail entity representing one of the triples, and \( \mathbf{v}_h, \mathbf{v}_r, \) and \( \mathbf{v}_t \) represent the vector representation of the corresponding head entity, relation, and tail entity. Next, the triplet embedding vectors are concatenated into sequences \( x \), represented as shown in Equation (3).

\[
x_i = [\mathbf{v}_h, \mathbf{v}_r, \mathbf{v}_t]
\]

Transformer models have been widely used in computer vision, computer recommendation, and natural language processing. The previous model [34] proves that the Transformer mechanism can more fully apply GPU parallel computing, and the Transformer can capture more sequence signals. The main structure is shown in Figure 3. This module first needs to obtain the sequence \( z^0 = \{z^0_1, z^0_2, z^0_3, \ldots, z^0_k\} \) through the sequence \( \{x_1, x_2, x_3, \ldots, x_k\} \), as shown in Equation (4).

\[
z^0_i = \mathbf{x}^{\text{input}}_i + \mathbf{x}^{\text{PE}}_i,
\]

where \( \mathbf{x}^{\text{input}}_i \) represents the vector representation of the sequence and \( \mathbf{x}^{\text{PE}}_i \) represents the corresponding position embedding vector of the sequence triplet. After obtaining the zeroth layer sequence \( z^0 \) representation, this chapter uses the Transformer encoder to obtain a new sequence, \( z^L = \{z^L_1, z^L_2, z^L_3, \ldots, z^L_k\} \). The specific representation is shown in Equation (5).

\[
z^L_i = \text{Transformer} - \text{Encoder}(z^{L-1}_i), L = 1, \ldots, K,
\]

where \( z^L_i \) represents the \( L \)-th hidden vector of \( z^0_i \). The multi-head attention mechanism is used in the Transformer encoder, and the calculation formula is shown in Equations (6) and (7).

\[
\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_n)W^O,
\]

\[
\text{head}_i = \text{Self - Attention}(QW^Q_i, KW^K_i, VW^V_i),
\]

where \( \text{head}_i \) represents the single-head self-attention mechanism, and \( n = 8 \) in the method in this chapter. The multi-head attention mechanism is composed of \( n \) single-head self-attention mechanisms. The parameter matrix \( W^Q_i, W^K_i, W^V_i \in R^{d_{\text{model}} \times d}, W^O \in R^{d_{\text{model}} \times d}, \) \( d_k = d_v = \frac{d_{\text{model}}}{n} \) in Formula (7). \( Q, K, \) and \( V \) represent the three-element Query, Key, and Value of the self-attention mechanism.
Through the above process, this module can obtain the weighted implicit features of the entity’s one-hop neighbors through the Transformer encoder, as shown in Equation (8).

\[ H_{e^i} = \text{Substr} \left( \sum_{j=2}^{n} \alpha_i \times z^i_j \right), \]  

(8)

Among them, \( z^i_j \) represents the hidden vector of the ith hop adjacent entity of the triple vector \( x_i \), and the substring function indicates that the final triple vector \( (h_i, r, t_i) \) of the source entity is intercepted to obtain the hidden feature vector of the high-order domain aggregation of the source entity. \( \alpha_i \) is the attenuation attention coefficient, which represents the attenuation degree of the i-th hop neighbor of the source node \( e_k \), as shown in Equation (9).

\[ \alpha_i = 1/\prod d_i , \]  

(9)

where \( d_i \) represents the distance between the adjacent entity and the source entity.

4.2. Neighbor Encoder

The goal of this module is to obtain implicit features through the entity enhancement encoder to update the feature vector of the entity. The schematic diagram is shown in Figure 4. First, we concatenate the feature vectors \( H_{t_k} \) of the triples, and then obtain the high-order domain aggregated information embedding of the source entity through the Transformer encoder between each layer and the hierarchical attention mechanism between different layers. Finally, the entity feature vector with more knowledge is obtained by adding the high-level domain aggregation information to the explicit feature information of the source entity.

Figure 3. The main structure of the entity-enhanced encoder.

Figure 4. The main structure of the neighbor encoder.
This paper first obtains the high-order domain aggregated feature vectors of tail entities, as shown in Equation (10).

\[ v'^t_k = v_t + H^t_k, \]

(10)

In vector weighting, in order to prevent the triple feature vector \( \{v_{h\text{'}}, v_{e\text{'}}, v_{t\text{'}}\} \) from overfitting, the head entity, relation, and tail entity feature vectors are generalized using the dropout [35] method, and the expression \( C_{h\text{'}, e\text{'}, t\text{'}} \) is shown in Equation (11).

\[ C_{h\text{'}, e\text{'}, t\text{'}} = W (v_{h\text{'}} \oplus v_{e\text{'}} \oplus v_{t\text{'}}) + b_c. \]

(11)

Among them, \( W \in \mathbb{R}^{d \times 3d}, b_c \in \mathbb{R}^d \) is the parameter that the method in this chapter needs to learn, and \( \oplus \) represents the concatenation of vectors.

The neighbor encoder proposed in this paper resembles GCN [22] and makes better on HAF. This paper uses a shared kernel \( \{W_c, b_c\} \) to encode different entity neighbors. The method in this chapter considers the multi-hop propagation of information but is set within the range of length 3 from the source entity. Because the propagation distance of information is too small, the methods will lose a large amount of hidden properties. If the propagation distance is too large, the scope of application of the model will be reduced, and the weight of domain information will also be reduced.

4.3. Noise Checker

The goal of this module is to perform a noise entity inspection and filtering on the candidate entity set. The main structure is shown in Figure 5. For each relation \( r \) and the corresponding real triples \( (h_0, r, t_0) \) in the embedding vector space, this module first uses all candidate entities as tail entities to construct reference triples \( \{h, r, t^\text{'}_k\} \in C_{h, r, t} \), and then takes the query vector \( v_q \) of the real triples as the core, and the reference triplet vector \( v_{R_k} \) as the satellite node. A vector of tuples for the satellite nodes. After forming a galaxy-like noise map, the module uses the distance of the reference triplet to the query triplet vector as a plausibility score. The larger the rationality score, the higher the rationality of the candidate entity; otherwise, the higher the noise of the candidate entity. The specific formula is shown in Equation (12).

\[ \text{Score}_i = \sqrt{\frac{1}{n+1} \sum_{j=0}^{n} (\text{Query}_j - \text{Reference}_j)^2} \]

(12)

The \( \text{Score}_i \) represents the rationality score of the \( i \)-th candidate entity, and \( \text{Query}_i \) and \( \text{Reference}_i \) represent the elements of the \( i \)-th dimension of the query vector \( v_q \) and the reference vector \( v_{R_k} \), respectively. After obtaining the rationality score, the model selects the top \( n \) candidate entities as the candidate set for model training through the score, where \( n \) is set to 20.
4.4. Matching Processor

The goal of this module is to perform similarity matching between the previously updated triples and the tested candidate entities. The main structure is shown in Figure 6.

LSTM is used to obtain the similarity score of candidate entities as follows:

\[
S'_{few} = \text{Enhanced} - \text{Encoder}(S_{few}),
\]

\[
hide_{k+1}, c_{k+1} = \text{LSTM}(q, \langle \text{hide}_k \oplus S'_{few}, c_k \rangle),
\]

\[
hide_{k+1} = \text{hide}_{k+1} + q,
\]

\[
\text{score}_{k+1} = \left( \text{hide}_{k+1} \cdot S'_{few} \right) / \left( \| \text{hide}_{k+1} \| \times \| S'_{few} \| \right)
\]
In the above formula, \( q = v_h \oplus v_r \oplus v_t \) represents the vector representation of the query triplet, \( S_{fow} = v_{h_i} \oplus v_{r_i} \oplus v_{t_i} \) represents the concatenation of the vectors, and \( \text{hide}_k \) represents the hidden vector of the LSTM. \( \text{Score}_k \) represents the similarity score between the source entity and the target entity after looping \( k \) times. Score ranking is used to predict the outcome of the model.

4.5. Model Training

The training goal of this paper is to conduct model inference through positive samples and negative samples. For each query relation \( r \) and reference triplet \( (h_a, r, t_a) \), negative samples \( \left\{ \left(h_i, r, t_i\right) \mid \left(h_i, r, t_i\right) \notin G \right\} \) are constructed by polluting the tail entities of the positive samples \( \left\{ \left(h_i, r, t_i\right) \mid \left(h_i, r, t_i\right) \in G \right\} \). Learning the previous embedding-based model, the model is optimized using a hinge loss function as shown in Equation (14).

\[
I_\beta = \max(0, \beta + S_n - S_p),
\]

among them, \( S_p \) represents the score obtained by the inference model for positive samples, \( S_n \) represents the score obtained by the inference model for negative samples, and the parameter \( \beta \) is the hyperparameter to be adjusted. For each training time slice, we first sample a task relation \( T_r \) from the meta-training set, then take one triple from all known triples as the reference triple and a batch of other triples as the positive query triple.

In summary, the model algorithm is defined as shown in Algorithm 1. In Algorithm 1, steps 5 to 7 are to enhance the representation of the head and tail entities of the sample (see Sections 4.1 and 4.2). Next, steps 8 to 10 proceed to construct positive and negative sample triples. Step 11 shows that the noise checker filters unreasonable candidate entities (see Section 4.3). Step 12 calculates the matching score for the reference triplet (see Section 4.4). Finally, steps 13 and 14 perform parameter training and updating (see Section 4.5).

---

**Algorithm 1. AAF**

**Input:**
(a) Meta-training task set \( T_{\text{meta-training}} \)
(b) DisMult Knowledge Graph embedding,
(c) Initial parameters \( \theta \) of the metric model,
(d) Sample size \( F \)

**Output:**
Save the optimal model parameters \( W, b_c, \theta \);
1: for \( \text{epoch} = 0 \) to \( M - 1 \) do
2: Shuffle the tasks in \( T_{\text{meta-training}} \);
3: for \( T_r \) in \( T_{\text{meta-training}} \) do
4: Sample \( F \) triples as the reference
5: for \( f \) in \( F \) do
6: Enhanced embedding vector representation of head and tail entities
7: end for
8: Processing query triples through an adaptive weight network
9: Sample a batch \( B^+ \) of query triples
10: Pollute the tail entity of query triples to get \( B^- \)
11: Noise checker filters candidate entities
12: Calculate the matching scores
13: Calculate the batch loss $\Lambda = \sum B \ell_{\theta}$
14: Update $\theta$ using gradient
15: end for
16: end for

5. Experiments
In this section, in order to estimate the performance of the proposed method and test the stability of the components in the model, we have performed a large number of experiments.

5.1. Dataset
To achieve the same experimental environment as the baseline model, we conduct extensive experiments on the classic knowledge graph dataset NELL-One. NELL-One is a subset of the NELL dataset [36]. The NELL dataset uses a framework to continuously extract information from the WEB and then performs data cleaning, data disambiguation, and data integration. The NELL-One dataset first expands the number of samples by adding reverse triples to all triples based on the source dataset and then selects relationships with more than 50 but less than 500 triples as a small-sample learning task dataset, which has 68,545 entities, 358 relations, and 181,109 triples. The NELL-One dataset assigns tasks to 51/5/11 as the training set/validation set/test set. The triplet relationship that does not meet the selection requirements of the small-sample learning task is called the background relationship, and the triplet that does not meet the conditions is used as the background relationship. Background image for model training.

5.2. Baseline Methods
We use the following two types of methods as baseline methods for comparison:

Based on embedding methods. Traditional embedding models learn relation-entity embeddings by modeling relational paths of knowledge graphs. This paper adopts four embedding-based methods: DistMult [15], ComplEx [16], RotatE [17], and TransE [18]. This paper uses a PyTorch-based implementation [7] for comparison. Since RESCAL uses matrices to represent relations, we employ mean pooling on these matrices to obtain 1-D embeddings. Entities and relationships are represented by one-dimensional vectors of TransE and DistMult, so we do not need to make any more modifications. For the ComplEx method, this paper combines the real and imaginary parts of the original method into an embedding vector.

Few-shot relationship learning method. There are two main types of neighbor-based encoders, GMatching and FSRL. The former considers that the weights of the adjacent entities of the source entity are the same and sets the neighbor vector to 0 during model training. The latter considers the influence of adjacent entities on the basis of the former and uses different weights for adjacent entities to improve the information of the source entity. The encoders based on meta-learning are the MetaR method and the HAF method; the Meta model tries not to rely on background graph knowledge, and two new meta-information core optimization strategies are proposed. The above centralized models do not consider the weight effect of the cascade and do not adopt dynamic representation for the influence of adjacent entities. The HAF model uses a self-attention mechanism and an attenuated attention mechanism to obtain higher-order domain aggregated vectors of source entities.

5.3. Experimental Design
The pre-trained model selects the DistMult model. The embedding dimension of the NELL-One dataset is 100, so the dimension of the hidden state of the LSTM is 200. The
optimization of model parameters uses the Adam optimizer. The trained model is applied to the validation task every 1000 time slices, compares the results of each training of the model with the best results, and then saves the best results of model training. Among them, the best result of Hits@10 is taken as the optimal result. The model in this paper is implemented using the PyTorch framework, and the GPU is used for parallel computing. The number of adjacent entities is set to 50, the learning rate is 0.001, and the first value of the parameter is 5.0.

5.4. Evaluation Metrics

We use the two most common evaluation metrics: mean reciprocal rank (MRR) and hyperlink-induced topic search (Hits@K), as evaluation metrics to evaluate the accuracy of model training. MRR is an international mechanism for evaluating search algorithms. The higher the rank of the correct answer, the higher the value of MRR. Hits@K is used to evaluate the average proportion of connection prediction rankings less than K in the knowledge graph. The higher the number of correct answers, the higher the value of Hits@K. In this paper, K is equal to 1, 5, and 10, respectively.

5.5. Results

This paper conducts extensive experiments to test the link prediction of the models to demonstrate the accuracy of the method, and all models achieve the highest results in the case of one reference sample (1-shot) and five reference samples (5-shot) on the NELL-One dataset. The performance is shown in Table 1. In the experimental results, the underlined numbers represent the best results of the GMatching model, and the bold numbers represent the best results of all methods. From Table 1, we can see that the highest performance of the method in this paper is higher than the experimental results of the baseline model in most cases, and in the single-sample scenario, the experimental results of the model in this paper are better than the baseline model. In the small-sample scenario, the MRR and Hits@1 results of our method are slightly lower than those of the third chapter method, but the result of Hits@10 is 3.5% higher than that of the third chapter method. Table 2 shows the average experimental results of one sample and five samples on the NELL-One dataset for GMatching, HAF, and our method. The average experimental results of the method in this paper are 1.5%–5.9% higher than those in HAF in the single-sample experimental scenario and 3.2% higher than the method in HAF Hits@10 in the small-sample task scenario.

Table 1. The 1-shot and 5-shot experimental results on the NELL-One dataset. Underline represents the best experimental result of GMatching, and the numbers in bold represent the best experimental results of all methods.
The experimental results on the NELL-One dataset are shown in Table 1. The performance of the method in this paper is 9.7% higher than that of the GMatching model in the single-sample task scenario and 8.1% higher than that of the GMatching model in the small-sample task scenario, which is better than the FSRL model. It is 3.7% higher, which is 6.3% and 3.5% higher than the HAF single-sample and small-sample task scenarios, respectively. To sum up, the experimental results show that our model outperforms the baseline model in both 1-shot and 5-shot scenarios. The experimental results can prove that:

1. The model in this chapter can adapt to experimental sample scenarios of different sizes and can handle single-sample training task scenarios well, with higher stability;
2. The method in this chapter mixes the debilitating attention mechanism and the Transformer’s high-order domain aggregated entity-enhanced encoder, which is beneficial to solving the sequence relationship of different training task scenarios;

In order to improve the adaptability of the model and reduce the computational consumption of the model in real-world applications, our model saves the training results with the highest Hits@10 during the training process for inferring incomplete facts later.

### 5.6. Ablation Analysis

The main improvement of the method in this paper is the entity augmentation encoder, which mainly includes Transformer and hierarchical attention mechanism. The Transformer mechanism has been widely proven to be effective for weights. In order to further verify the effectiveness of the hierarchical attention mechanism, further ablation analysis is performed in this paper. The experimental results are shown in Table 3. In Table 3, Two level represents the model using two-layer aggregation layers in related fields, Three level represents the model using three-layer aggregation layers in related fields, and the underlined numbers represent the best results with the highest performance in the two-layer and three-layer models. Bold numbers represent the best results for the average performance of the two models.

<table>
<thead>
<tr>
<th>NELL-One</th>
<th>MRR</th>
<th>Hits@1</th>
<th>Hits@5</th>
<th>Hits@10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-Shot</td>
<td>5-Shot</td>
<td>1-Shot</td>
<td>5-Shot</td>
</tr>
<tr>
<td>GMatching(mean)</td>
<td>17.6</td>
<td>-</td>
<td>11.1</td>
<td>-</td>
</tr>
<tr>
<td>HAF(mean)</td>
<td>19.5</td>
<td>19.9</td>
<td>13.0</td>
<td>13.3</td>
</tr>
<tr>
<td>Ours(mean)</td>
<td><strong>22.5</strong></td>
<td><strong>20.0</strong></td>
<td><strong>14.5</strong></td>
<td>12.0</td>
</tr>
</tbody>
</table>

The highest and average performance of the model with three aggregation layers in Table 3 is higher than that of the model with two aggregation layers. The experimental results show that when the number of aggregation layers is larger within the threshold,
the experimental results in this paper are better. The threshold value in this paper is 3 because the computational complexity increases exponentially with the larger the number of aggregation layers, and the noise error becomes larger. In summary, the effectiveness of the hierarchical attention mechanism of our method can be demonstrated.

5.7. Results of Different Relations

The previous experimental results can prove that the overall inference performance of the model in this paper is better. To further demonstrate the effectiveness of different relations, this section shows the performance of all relations on the NELL-One dataset, as shown in Table 4. Because the experimental results of HAF are better than those of other baselines, HAF is used as the baseline model in this section. Rid represents the id of the triple relation, #Candidate represents the number of candidate sets for the relation, and the bold numbers represent the best result of the relation.

Table 4. Performance representation of different relationships in the test dataset. Bold numbers denote the best results of the model.

<table>
<thead>
<tr>
<th>Rid</th>
<th>#Candidate</th>
<th>1-Shot HAF</th>
<th>1-Shot Ours</th>
<th>5-Shot HAF</th>
<th>5-Shot Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>123</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
<td>1.000</td>
</tr>
<tr>
<td>2</td>
<td>299</td>
<td>0.176</td>
<td>0.515</td>
<td>0.176</td>
<td>0.426</td>
</tr>
<tr>
<td>3</td>
<td>786</td>
<td>0.686</td>
<td>0.696</td>
<td>0.686</td>
<td>0.700</td>
</tr>
<tr>
<td>4</td>
<td>1084</td>
<td>0.768</td>
<td>0.863</td>
<td>0.863</td>
<td>0.853</td>
</tr>
<tr>
<td>5</td>
<td>2100</td>
<td>0.892</td>
<td>0.794</td>
<td>0.853</td>
<td>0.843</td>
</tr>
<tr>
<td>6</td>
<td>2160</td>
<td>0.205</td>
<td>0.304</td>
<td>0.257</td>
<td>0.270</td>
</tr>
<tr>
<td>7</td>
<td>2222</td>
<td>0.619</td>
<td>0.691</td>
<td>0.669</td>
<td>0.777</td>
</tr>
<tr>
<td>8</td>
<td>3174</td>
<td>0.585</td>
<td>0.741</td>
<td>0.552</td>
<td>0.604</td>
</tr>
<tr>
<td>9</td>
<td>5716</td>
<td>0.220</td>
<td>0.379</td>
<td>0.198</td>
<td>0.367</td>
</tr>
<tr>
<td>10</td>
<td>10569</td>
<td>0.009</td>
<td>0.038</td>
<td>0.012</td>
<td>0.053</td>
</tr>
<tr>
<td>11</td>
<td>11618</td>
<td>0.028</td>
<td>0.169</td>
<td>0.035</td>
<td>0.162</td>
</tr>
</tbody>
</table>

It can be seen from Table 4 that most of the relational performances of the methods in this chapter are higher than the baseline models. The fifth relation in the table is slightly lower than HAF in the single-sample and small-sample task scenarios, and the fourth relation is also slightly lower than HAF in the small-sample task scenario, but the stability is higher than that of the HAF model, as shown in Figures 7 and 8 shown. Therefore, the experimental results in this paper are closer to the real results. Among them, the vertical axis in Figures 7 and 8 is the performance value of Hits@10, the horizontal axis is the number of training times of the model, the red broken line represents the performance value of the HAF model, and the blue represents the performance value of the method in this paper.
Figure 7. The 1-shot learning curve on NELL-One.

Figure 8. The 5-shot learning curve on NELL-One.

6. Conclusions

This paper introduces a few-shot relation learning framework with adaptive weights for novel fact prediction of long-tailed entities under different sample sizes. Our model is overall optimized using Transformer and Hierarchical Attention Mechanism Entity Augmentation Encoder, Neighbor Encoder, Noise Checker, and Matching Processor. Extensive experiments on public datasets show that our model outperforms baseline methods across different sample sizes. The effectiveness of our model components is then demonstrated using ablation studies. Our future work may consider adding multi-step queries to our model to improve the ability of model information mining or adding time information to consider a small-sample knowledge graph reasoning model on time series.

Author Contributions: Conceptualization, R.M. and Z.L.; methodology, Z.L.; software, Z.L.; validation, Z.L. and Y.M.; formal analysis, H.W.; investigation, M.Y.; resources, Z.L.; data curation, Z.L.; writing—original draft preparation, Z.L.; writing—review and editing, Z.L.; visualization, Z.L.; supervision, L.Z.; project administration, Z.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China (61906030), the Science and Technology Project of Liaoning Province(2021JH2/10300064), the Natural Science Foundation of Liaoning Province(2020-BS-063) and the Youth Science and Technology Star Support Program of Dalian City (2021RQ057)

Data Availability Statement: Data available on request due to privacy.
Acknowledgments: This work is supported by the National Natural Science Foundation of China (61906030), the Science and Technology Project of Liaoning Province (2021JZ2910300064), the Natural Science Foundation of Liaoning Province (2020-BS-063) and the Youth Science and Technology Star Support Program of Dalian City (2021RQ057).

Conflicts of Interest: The authors declare no conflict of interest.

References


