Ruleformer: Context-aware Rule Mining over Knowledge Graph

Zezhong Xu[1] **, Peng Ye**[1] **, Hui Chen**[3]**, Meng Zhao**[4]
Huajun Chen[1,2]**, Wen Zhang**[1*]
[1]Zhejiang University & AZFT Joint Lab for Knowledge Engine, China
[2]Hangzhou Innovation Center, Zhejiang University
[3]Alibaba Group, [4]Huawei
{xuzezhong, yep, huajunsir, zhang.wen}@zju.edu.cn,
weidu.ch@alibaba-inc.com, zhaomeng57@huawei.com

Abstract

Rule mining is an effective approach for reasoning over knowledge graph (KG). Existing works mainly concentrate on mining rules. However, there might be several rules that could be applied for reasoning for one relation, and how to select appropriate rules for completion of different triples has not been discussed. In this paper, we propose to take the context information into consideration, which helps select suitable rules for the inference tasks. Based on this idea, we propose a transformer-based rule mining approach, **Ruleformer**[1]. It consists of two blocks: 1) an encoder extracting the context information from subgraph of head entities with modified attention mechanism, and 2) a decoder which aggregates the subgraph information from the encoder output and generates the probability of relations for each step of reasoning. The basic idea behind Ruleformer is regarding rule mining process as a sequence to sequence task. To make the subgraph a sequence input to the encoder and retain the graph structure, we devise a relational attention mechanism in Transformer. The experiment results show the necessity of considering these information in rule mining task and the effectiveness of our model.

1 Introduction

People built different Knowledge Graphs, such as Freebase (Bollacker et al., 2008), to store complex structured information and knowledge about abstract and real world. The facts in KG are usually represented in the form of triplets, e.g., *(New York, isCityOf, USA)*. KGs have been widely applied in various intelligent systems such as question answering (Yasunaga et al., 2021; Chen et al., 2022), recommender system(Guo et al., 2020; Wong et al., 2021; Zhang et al., 2021) and zero-shot learning(Geng et al., 2022; Chen et al., 2021b).

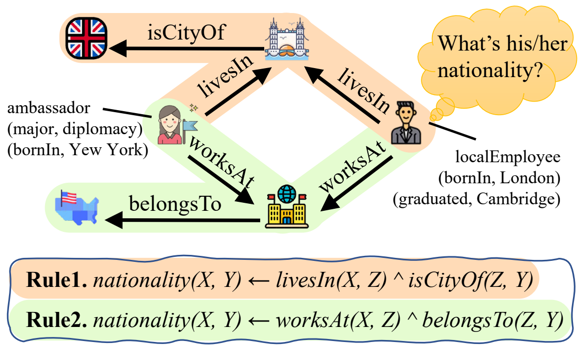
isCityOf
livesIn
livesIn
What's his/her nationality?
ambassador
(major, diplomacy)
(bornIn, Yew York)
worksAt
worksAt
localEmployee
(bornIn, London)
(graduated, Cambridge)
belongsTo
Rule1. *nationality(X, Y) ← livesIn(X, Z) ^ isCityOf(Z, Y)*
Rule2. *nationality(X, Y) ← worksAt(X, Z) ^ belongsTo(Z, Y)*

Figure 1: Rules in different context.

Although these KGs already contain a large number of relations and entities, they still suffer from the incompleteness of facts, whether constructed automatically or manually. In order to further expand KGs, many reasoning methods have been proposed to automated fact exploration, such as knowledge graph embedding (KGE) (Bordes et al., 2013; Trouillon et al., 2016; Dettmers et al., 2018; Sun et al., 2019; Zhang et al., 2019b), graph neural networks (Schlichtkrull et al., 2018), and rule mining methods (Ortona et al., 2018; Galárraga et al., 2013; Yang et al., 2017; Sadeghian et al., 2019; Zhang et al., 2019a). Compared with deep learning approaches like KGE, rule mining methods are preferred due to their interpretability for reasoning and robustness for domain knowledge transfer. To mine the structure and confidence of rules at the same time in a fast way, differentiable rule learning methods(Yang et al., 2017; Sadeghian et al., 2019) are introduced and attract many research interests.

The learning targets of existing methods is determining the confidence and structure of rules, according to which reasoning tasks are conducted. However, the uniform way of applying rules ignores the context of specific triplets, for which rules should be applied in different order.

Specifically, for a query $(h, r, ?)$, there might be multiple rules of relation r (not only one) that

[*] Corresponding Author.
[1]Source code of Ruleformer is available at https://github.com/zjukg/ruleformer.

could be used to get the target answer. For a specific query triplet, not all rules of r are convincing for reasoning, but how to select appropriate rule has not been studied. For example, to infer the missing fact, *(*`localEmployee/ambassador`*, nationality, ?)* as Figure 1 illustrated, previous rule mining methods only rely on the head relation *nationality*, so rules of *nationality* will be used in a fixed order. The two queries will get the same results, and whatever the order is, one of answers will be wrong. More precious rules can be observed if we pay attention to the head entity's identity, which is `localEmployee/ambassador`. Based on the observation, when the head entity is a `localEmployee`, the rule *nationality(X, Y)* $\leftarrow$ *livesIn(X, Z)* $\wedge$ *isCityOf(Z, Y)* may derive more accurate result. In another case, if we know the head entity is an `ambassador`, then rule *nationality(X, Y)* $\leftarrow$ *worksAt(X, Z)* $\wedge$ *belongsTo(Z, Y)* will be a better choice for inferring. This example shows that the order and choice of rules to be applied for reasoning will significantly affect the results, and context information could help determine proper order.

In this paper, we investigate making rule mining methods not only learn confidence and structures of rules but also learn to choose suitable rules based on the context of the head entity in completion tasks, which is challenging because of the diversity of context. We propose a Transformer-based model to aggregate the context around the head entity because of its excellent performance on information interaction. Essentially, the model consists of two blocks, 1) an encoder block aggregating the information of the head entity's subgraph and completing the information intersection of context, 2) a decoder block utilizing the aggregated entity embedding to generate the probability of relations for each step in differentiable rule mining process. Since Transformer framework is a sequence to sequence model, we design a converter that can turn graph structure into sequence. Moreover, to maintain the information of subgraph, we modify the attention mechanism of Transformer by adding the relation information to the attention calculation process, which we call relational attention mechanism.

Experimentally, we evaluate our model on several datasets (*UMLS* (Kok and Domingos, 2007), *FB15K-237* (Toutanova et al., 2015), *WN18RR* (Dettmers et al., 2018)) on link prediction and rule parsing task. The improvement in both experiments demonstrate the effectiveness of Ruleformer. A case study is also analyzed, and the case proves our assumption and the ability of Ruleformer to select suitable rules for different triples.

In summary, our contributions are as follows:

- We draw attention to the problem of mining and applying suitable rules depending on the specific context for reasoning task in KG.
- We propose a new model, Ruleformer, that can aggregate the information of subgraph and use the context to support the reasoning process.
- The experiment results prove that our model outperforms existing rule mining methods on link prediction task and rule quality assessment. It successfully selects suitable rules according to the exploitation of context.

2 Related work

2.1 Rule Learning Methods

The problem of learning rules over KG can be seen as a type of statistical relational learning (Koller et al., 2007). AMIE (Galárraga et al., 2013) concentrates on association rule mining with three operations, including dangling atom, instantiated atom and closing atom that add different type of atoms to incomplete rules and uses pre-defined evaluation metrics to prune incorrect rules. AMIE+ (Galárraga et al., 2015) revises the rule extending process and improves evaluation method based on AMIE.

Anyburl (Meilicke et al., 2019) proposed an framework that can mine rules in an effcient way. Based on the randomly sampled path, it replace some entities with variables and get rules.

Rudik (Ortona et al., 2018) can mines positive and negative rules. The positive rules can be used to infers new facts in KG, and the negative rules are useful for other tasks, like detecting erroneous triples.

Generally, conventional symbolic-based rule learning methods are built on effective search strategy, pruning techniques and pre-defined static evaluation indicators. The inference processes are transparent while they may suffer from large search space.

More recently, differentiable rule learning methods based on TensorLog (Cohen, 2016) are proposed, which can learn the confidence and structure of rule at the same time. Neural-LP (Yang et al.,

2017) use RNN to generate the possibilities of different relation for each step, and the parameters can be optimized in a differentiable way. Based on Neural-LP, DRUM (Sadeghian et al., 2019) is proposed, which use low rank approximation to get better results. Neural-Num-LP (Wang et al., 2019) extends Neural-LP to learn the numerical rules. Neural Logic Inductive Learning (Yang and Song, 2020) uses transformer structure to get the non-chain-like rules which To extend the diversity of mined rules.

Whether it's pure-symbolic or neural-symbolic method, none of these models consider the problem of utilizing the information about the head entity during the process of mining or using rules.

2.2 Embedding-based Models

A lot of previous works (Bordes et al., 2013; Trouillon et al., 2016; Dettmers et al., 2018; Sun et al., 2019) concentrate on the embedding-based paradigm. Most of them design a scoring function to get a value for a triplet in the embedding space. Despite their simplicity, embedding-based models achieved good performance on reasoning. These works mostly rely on simple triplets, which means they ignore the environment of the entities and relations.

Some other embedding-based methods consider context information during inferring process. PTransE (Lin et al., 2015) uses the path embedding from the source entity to target entity and the relation embedding to train the model jointly. (Das et al., 2017) further considers entities and entity type in the path. Except concentrating on the path, some methods (Schlichtkrull et al., 2018) utilize information from context with graph neural networks (GNN). HittER (Chen et al., 2021a) uses hierarchical Transformers to make contextualization based on a source entity's neighborhood because they think the network architecture of GNNs is too shallow.

These approaches prove contextual information which plays an important role in reasoning, and this aspires us to consider the environment of source entity in rule mining to some extent. Although the performance on link prediction task is good, these works mostly are based on vector computation, so the symbolic meaning and interpretability are missing while rule mining methods perform better in this regard.

3 Methodology

Knowledge Graph $\mathcal{G}$ is composed by a set of triplets like $\{(e_s, r, e_t) | r \in \mathcal{R}, e_s, \in \mathcal{E}, e_t \in \mathcal{E}\}$, where $\mathcal{E}$ is a countable set of entities and $\mathcal{R}$ is a set of relations, respectively. For the task of rule mining, all the triplets that belong to $\mathcal{G}$ are given to the model, and for each head relation r, the model is supposed to find meaningful rules that are interpretable and understandable for humans. The rule we want is in the following form:

$$r(X, Y) \leftarrow r_1(X, Z_1) \wedge ... \wedge r_T(Z_{T-1}, Y)$$

where T is the length of rule, r and r_i are relations belonging to $\mathcal{R}$, X, Y and Z_i are variables that can be replaced by specific entities and $r(X, Y)$ is a triplet. The triplet on the left of the arrow is called *head* of rule and the right part of the arrow is called rule *body*.

In this section, to provide an intuition about each part of our model, we introduce the details about Ruleformer. Firstly, an encoder is designed for converting the subgraph structure to a sequence, so that the context information of head entity can be input to the Transformer framework with relational attention mechanism which is used to retain the structure information of the graph completely. Then we present how our model is deployed to utilize the output of encoder to support the rule mining process and generate the target sequence that represents rule body. Finally, we propose a case-based rule parsing algorithm to get symbolic rules from the parameters.

3.1 Ruleformer

Now we introduce the subgraph encoding process of the model, and Figure 2 shows the details. The basic idea of our approach is to find rules for the same head relation and let the model has the ability to choose the suitable rule when the contextual environment of the head entity is different.

For a query (h, r, t), we assume that the subgraph around the head entity h in KG contains the potential knowledge needed for understanding the environment, so we first extract the subgraph around h. More precisely, let $\mathcal{S}_k(h)$ be the subgraph which contains the k-hop (shortest distance to h) neighbor of h in the KG and the set of edges connecting these entities.

As we stated before, KG is organized as a graph structure, while Transformer (Vaswani et al., 2017) is a seq2seq model, so we need to transfer

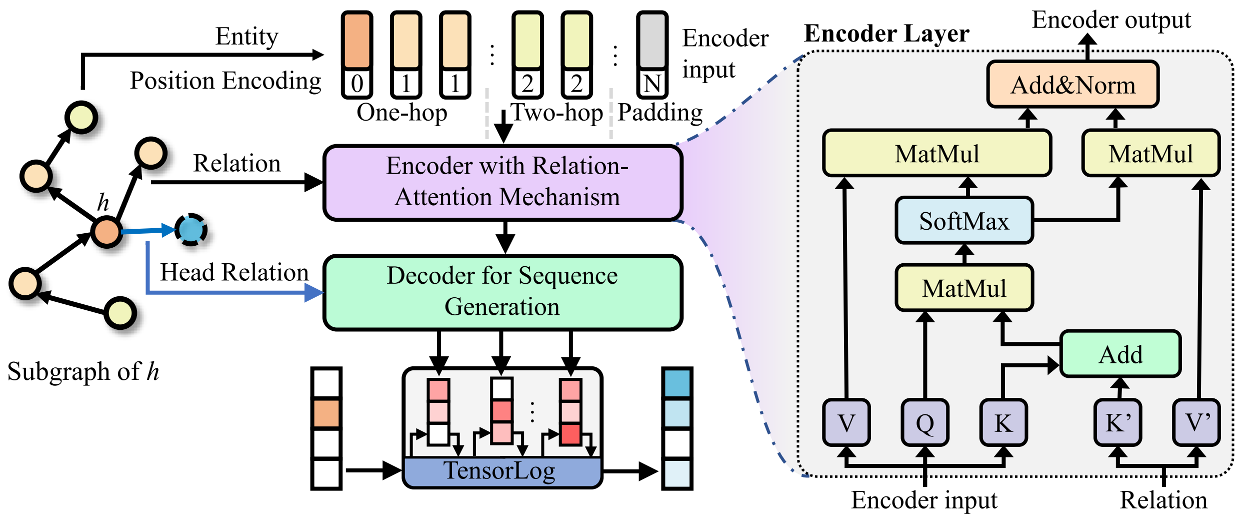

Figure 2: The framework of Ruleformer and details about relational attention mechanism.

the graph to a sequence. Specifically, the nodes in the subgraph are tokenized into a sequence, $S_{node} = [e_1, e_2, ...e_{num}, ...blank]$, where num is the number of entities in the subgraph, and a special token *blank* is needed for padding. As shown in the Figure 2, each node in the subgraph is extracted, and mapped to the initial embedding for entities. The shortest distance to the head entity h is also added as position embedding.

Meanwhile, the distribution of entity occurrence is heavy tailed and hence it is hard to learn appropriate representations for each entity. To alleviate this problem, we consider using the type of relations to help represent the entity. For each relation r, we define two randomly initialized embedding r^{dom} and r^{ran} representing the domain and range embedding of r, which can be learned during the training process. An entity e gets its representation x_e by addition of the type embedding of relations connecting to the entity and a randomly initialized embedding y_e as follows:

$$x_e = \sum_{i=1}^{|\mathcal{R}|} b_i^{dom} r_i^{dom} + \sum_{i=1}^{|\mathcal{R}|} b_i^{ran} r_i^{ran} + y_e \quad (1)$$

where b_i^{dom} and b_i^{ran} are the parameters which are determined by the numbers of different relation types after normalization, and can be given as:

$$b_i^{dom} = \frac{n_i^{dom}}{\sum_{j=1}^{|\mathcal{R}|} n_j^{dom}}, \quad b_i^{ran} = \frac{n_i^{ran}}{\sum_{j=1}^{|\mathcal{R}|} n_j^{ran}} \quad (2)$$

where n_i^{dom} and n_i^{ran} is the number of each type relation connecting to and connected from e, specifically.

Thus, the node sequence S_{node} in mapped to a sequence of embedding $S_e = [x_1, x_2, ...x_{num}, ...blank]$. Note that the above steps only use the entities in the subgraph without the edges between them. In order to utilize the edge information, we introduce relational attention mechanism. Different from the basic way of attention calculation in Transformer, we modify the steps to compute the attention a_{ij} between entity i and j with relations connectingin the following way:

$$a_{ij} = \frac{(x_i W^Q)(x_j W^K + \sum_{r=1}^{|\mathcal{R}|} k_r x_r W^{K'})}{\sqrt{d_k}} \quad (3)$$

where x_i and x_j are the embedding of entity i and j, respectively. x_r is the embedding for relation r which is different from r^{dom} and r^{ran}. W^Q and W^K are the query and key matrices for entity, while $W^{K'}$ is the key matrix for relation. d_k is the dimension of W^K. k_r is defined as $k_r = 1$ if $r(e_i, e_j) \in \mathcal{G}$, else $k_r = 0$.

Then the normalization step for attention is executed:

$$\alpha_{ij} = \frac{\exp a_{ij}}{\sum_{k=1}^{n} \exp a_{ik}} \quad (4)$$

Similarly, we add the relation to the value computation step:

$$z_i = \sum_{j=1}^{n} \alpha_{ij} (x_j W^V + \sum_{r=1}^{|\mathcal{R}|} k_r x_r W^{V'}) \quad (5)$$

where W^V is the value matrix for entity and $W^{V'}$ for relation. By doing this, the information of relations has been inserted into the encoding process.

The relational attention mechanism can ensure sufficient information exchange between each entity because the relations that clearly exist will be emphasized, while the relations that do not appear in the incomplete KG but may be correct will also be reflected because of the interaction process.

With the encoder output sequence S'_e, the decoder block does most of the lifting to aggregate the information together with the head relation. As we mentioned before, the rule mining process can be regarded as a sequence generation problem. For each step, the decoder generates the most suitable relation from $\mathcal{R}$, until the length of decoder output sequence S_r reaches the rule length T.

Specifically, the rule sequence S_r input to decoder starts with the head relations r's embedding x_r, which means $S_r^0 = x_r$. After cross attention calculation with S'_e, the decoder gets a vector which implies the next relation. With this vector, an MLP function is deployed, and we can get the probability ω_t^i of relation r_i in step t, which will be used in the reasoning process. The relation with the highest probability is chosen as the next relation added to S_r.

$$\omega_{t+1} = MLP(CrossAttention(S_r^t, S'_e)) \quad (6)$$

where $\omega_{t+1} \in \mathbb{R}^{|\mathcal{R}|\times 1}$ could be interpreted as the probabilities of all relations in step t. Let r_{t+1} be the relation with the max probability, then $S_r^{t+1} = [S_r^t, x_{r_{t+1}}]$.

We repeat this step T times, and get the complete rule body with length T. Moreover, considering that rule length can't be limited to a definite number T, we add a special relation *self-loop* which connects entities with themselves, and we finally remove this relation from the rule body so that we can get rules with length less than T.

There is still a problem that it's different from the task of machine translation that we don't have labels to judge if the generated relation is an appropriate choice for each step. To overcome this issue, we deploy the Tensorlog framework here to get the prediction results like Neural-LP (Yang et al., 2017) and DRUM (Sadeghian et al., 2019) to indirectly supply supervisory signal. Each entity e_i is represented as a one-hot vector $\mathbf{v}^{e_i} \in \{0,1\}^{|\mathcal{E}|}$, and each relation r_k is represented as an adjacent matrix $\mathbf{M}^{r_k} \in \{0,1\}^{|\mathcal{E}|\times|\mathcal{E}|}$, where $\mathbf{M}_{ij}^{r_k} = 1$ if $r_k(e_i, e_j) \in \mathcal{G}$, else $\mathbf{M}_{ij}^{r_k} = 0$.

Via applying the entity vector and relation matrix, path queries could be answered by expanding path as follows:

$$\mathbf{v}' = \mathbf{v}^{e_i}\mathbf{M}^{r_k} \quad (7)$$

Note that $\mathbf{v}'$ is a multi-hot vector that refers to several entities, which means these entities are connected to e_i via relation r_k.

For step t, the probability ω_t of all relations, which is generated in Equation 6, are used by applying the above step in an indirect way as Equation 8. Let $\mathbf{z}_t \in \mathbb{R}^{|\mathcal{E}|\times 1}$ be the vector representing the probability of all entities in step t, and it is a one-hot vector that represents the head entity if $t = 0$. With $\mathbf{z}_{t-1}$ after $t-1$ step inference, $\mathbf{z}_t$ can be computed as follow:

$$\mathbf{z}_t = \mathbf{z}_{t-1} \times \sum_{i=1}^{|\mathcal{R}|+1} \omega_t^i \mathbf{M}^{r_i} \quad (8)$$

The special relation $self-loop$ with an identity adjacency matrix with $\mathbf{M}^{r_{|\mathcal{R}|+1}} = I_{|\mathcal{E}|}$ is also considered. With this relation, the model can mine rule with length shorter than the length T. ω_t^i represents the probability of the relation r_i as the relation in rules at step t.

Finally, we get the $\mathbf{z}_T$, which is the final result after T steps reasoning. For triplet (h, r, t), the reasoning score is the similarity between the predicted vector $\mathbf{z}_T$ and the target entity vector $\mathbf{v}$:

$$\phi(t|h,r) = \mathbf{v} \cdot log[\mathbf{z}_T, \gamma]_+ \quad (9)$$

where $[\mathbf{x}, \gamma]_+$ denotes the maximum value between each element of $\mathbf{x}$ and γ. The objective function Ruleformer is:

$$\min \left(- \sum_{(h,r,t)\in\mathcal{G}} \phi(t|h,r) \right) \quad (10)$$

3.2 Rule Parsing

To decode symbolic rules from Ruleformer, we propose a rule parsing algorithm using the parameters learned from training process. The basic idea is to select appropriate relations with high weight. With different triplets and the context information fed to the model, Ruleformer may output different parameters, so even for the same relation, the rules mined might be different. Specifically, for a query (h, r, t), we recover possible rules via parameters α. In each step, we choose relations whose weights are over the threshold, and we will check if the entities in the previous step are linked with this relation. By performing this step cyclically until rule length reaches the max length T, the confidence of each rule is computed by multiplying the weights of relations selected. output symbolic rules with high confidence. Finally, rules that may be useful in reasoning process for a triplet will be output. The detailed procedure is shown in Algorithm 1.

Then we summarize the number occurrences of each rule, and for each time a rule is applied, a confidence score will be calculated with ω and the final score is the average confidence of this rule in different cases. Specifically, a rule set is defined, and for each triplet (h, r, t), we apply Algorithm 1 to parse rules from it. Finally, the score will be calculated for all the rules in R.

Algorithm 1 Decode symbolic logical rules from model

```
Input: attention $\{\omega_t | t = 1, 2...T\}$ for each triplet
Initialize: $P = \{([P_r], [P_e], w)\}$ , $P_r = \emptyset$, $P_e =$ head entity, $w$ represents confidence;
Output: $R$
1:  for $t = 1 : T$ do
2:      // Scale the attention
3:      $\omega_t = \omega_t / max(\omega_t)$
4:      for $([P_r, P_e], w) \in P$ do
5:          for $\omega_t^{r_p} \in \omega_t > thr$ do
6:              // Expand a new path if possible
7:              for $n \in \mathcal{E}$ can be linked with $P_e[-1]$ via $r_p$ do
8:                  // Compute the new confidence
9:                  $w' = w \times \omega_t^{r_p}$
10:                 add $([P_r + r_p], [P_e + n], w')$ to $P$
11:             end for
12:         end for
13:         // Remove the old path
14:         remove $([P_r, P_e], w)$ from $P$
15:     end for
16: end for
17: for $([P_r, P_e], w) \in P$ do
18:     // $R[r, p_r]$ is a list stores confidence scores
19:     add $w$ to $R[r, p_r]$
20: end for
```

4 Experiment

4.1 Dataset

We conduct experiments on three different datasets which are introduced as follows, and Table 2 summarizes the data statistics. We count the average degree for each dataset, because the sparsity has an impact on the choice of subgraph.

- *UMLS* (Kok and Domingos, 2007): Unified Medical Language System, is a knowledge graph that brings together many health and biomedical vocabulary and standards.

- *FB15K-237* (Toutanova et al., 2015): This dataset is a subset from Freebase(Bollacker et al., 2008) and removes the inverse relation. It stores commonsense facts such as topics in movies, actors, awards, etc.

- *WN18RR* (Dettmers et al., 2018): WN18RR is a link prediction dataset created from WN18, which is a subset of WordNet. It's created to ensure that the evaluation dataset does not have inverse relation test leakage.

4.2 Experiment Setup

The experiments are implemented with Pytorch framework and are trained on RTX3090 GPU. ADAM optimizer is used for parameter tuning and the learning rate is set to 0.0001. Our model consists of the encoder and decoder block, which are two-layer Transformer and each layer has 6 heads by default. The dimension of entity is set to 200, as well as the position encoding dimension. We also try 6 heads and 3 layers in Transformer and larger dimension, which result in a little difference. Dropout is applied with a possibility $p = 0.1$. The γ used as threshold is set to be 10^{-20}.

Since the average degree is different for each dataset, to avoid the input sequence of entities being too long, the choice of subgraph is different, too. We extract one-hop subgraph for *FB15K-237*, two-hop subgraph for *WN18RR*. For *UMLS*, the shortest distance is the same as the rule length. Meanwhile, the maximum number of entities which is based on the sparsity of each dataset is different. Considering that relations with a large number of neighbors , like *hasGender*, don't contain much information relative to the huge number of entities, a max number of direct neighbor linked by one relation is set for each entity. For *UMLS*, we set the max number of context entities and max number of neighbors for each entity as 140 and 40 respectively. For *FB15K-237* and *WN18RR*, they are 70 and 40, and 40 and 10 respectively. If the number of context or neighbors exceeds our settings, we randomly select the same number of entities as the setting.

Methods	UMLS				FB15K-237				WN18RR			
	MRR	HIT			MRR	HIT			MRR	HIT		
		@1	@3	@10		@1	@3	@10		@1	@3	@10
TransE	.668	46.8	84.5	93.0	.294	-	-	46.5	.226	-	-	50.1
DistMult	.753	65.1	82.1	93.0	.241	15.5	26.3	41.9	.430	39.0	44.0	49.0
ComplEx	.829	74.8	89.7	96.1	.247	15.8	27.5	42.8	.440	41.0	46.0	51.0
ComplEx-N3	-	-	-	-	.370	-	-	**56.0**	**.480**	-	-	57.0
ConvE	.908	86.2	94.4	98.1	.325	23.7	35.6	50.1	.430	40.0	44.0	52.0
TuckER	-	-	-	-	**.358**	**26.6**	**39.4**	54.4	.470	**44.3**	48.2	52.6
RotatE	**.948**	**91.4**	**98.0**	**99.4**	.338	24.1	37.5	53.3	.476	42.8	**49.2**	**57.1**
PathRank	.197	14.7	25.6	37.6	.087	7.4	9.2	11.2	.189	17.1	20.0	22.5
Neural-LP(T=2)*	.751	63.0	84.7	94.0	.189	12.7	20.6	31.3	.371	35.9	37.4	39.6
Neural-LP(T=3)*	.735	62.7	82.0	92.3	.239	16.0	26.1	39.9	.425	39.4	43.2	49.2
DRUM(T=2)*	.791	64.5	92.7	96.8	.225	17.1	25.4	35.8	.379	36.8	38.5	40.9
DRUM(T=3)*	.784	64.3	91.2	97.2	.328	24.7	36.2	49.9	.441	41.2	45.6	51.6
M-Walk	-	-	-	-	.232	16.5	24.3	-	.437	41.4	44.5	-
Ruleformer(T=2)	.851	73.6	**96.6**	**98.8**	.237	17.4	25.7	36.0	.381	36.6	38.8	41.1
Ruleformer(T=3)	**.857**	**75.2**	95.8	98.4	**.342**	**25.5**	**37.4**	**51.3**	**.452**	**41.7**	**46.5**	**53.0**

Table 1: Link prediction results on dataset *UMLS*, *FB15K-237* and *WN18RR*. Note that for some algorithm, several entities may get the same score. Instead of computing the rank of the right answer as $m+1$ where m is the number of entities with higher possibilities, we select a random rank for the right answer among the entities with the same possibility. Some methods reported their results in the previous setup in their original paper, and we rerun these methods with the same evaluation process in our way. The results with [*] is reported with our evaluation protocol.

Datasst	UMLS	FB15K-237	WN18RR
#Triplet	5,960	310,116	93,003
#Entity	135	14,541	40,943
#Relation	46	237	11
#Avg.deg	88.3	42.7	4.5

Table 2: Knowledge graph datasets statistics.

4.3 Link Prediction

For the link prediction task, each triplet (h, r, t) in test dataset and its inverse triplet (t, r^{-1}, h) are given to the model with the tail entity masked, and the goal is to predict the masked tail entity among all entities. Each candidate entity in KG will get a score according to the inference result, which is used to sort them, and the ground truth entities are filtered out of the ranking list. Here, we adopt the evaluation metrics the same as previous works (Bordes et al., 2013), Hit@1, Hit@3, Hit@10, and mean reciprocal rank(MRR).

We compare our model with some embedding methods including TransE (Bordes et al., 2013), DisMult (Yang et al., 2015), ConvE (Dettmers et al., 2018), TuckER (Balazevic et al., 2019) and RotatE (Sun et al., 2019), and rule mining methods like PathRank (Lao and Cohen, 2010), Neural-LP (Yang et al., 2017), DRUM (Sadeghian et al., 2019) and M-Walk (Shen et al., 2018). Compared to embedding methods, rule mining methods can provide the interpretability, which is an advantage to pure embedding methods. The results are shown in the Table 1.

As the Table shows, our model outperforms the other rule-based approaches on three datasets. On *UMLS*, the competitive results demonstrates the effectiveness of rule mining approaches. Ruleformer offers an absolute improvement in hit@1 about 10.9% (relatively 16.9%) compared against baseline on *UMLS*. On dataset *FB15K-237* and *WN18RR*, our model also gets the best results on rule mining methods which improves the overall results about 1.4% and 1.1% (relatively 4.3% and 2.3%), respectively. Ruleformer achieves better performance compared to other rule mining proves the effectiveness of our method, and we think this improvement is because other rule-based methods don't consider the head entity and its subgraph and confirms the correctness of our hypothesis.

4.4 Quality Assessment of Minded Rules

In order to have an objective evaluation of the mined rules, we adopt the Standard Confidence (SC) (Galárraga et al., 2013) to assess the rules mined by different methods. Specifically, the average score of rules is calculated to show the quality. Given a list of rules which is ranked by their confidence calculated by their algorithm that is different from SC, we report the average SC of the top$K$ ($K = 50, 100, 200, 500$) rules. The rules of Neural-LP and DRUM are provided by their original code. The Standard Confidence $SC(\mathbf{B} \rightarrow r(X,Y))$ is calculated as follows:

$$\frac{\#(X,Y) : \exists Z_1...Z_m : \mathbf{B} \wedge r(X,Y)}{\#(X,Y) : \exists Z_1,\ldots,Z_m : \mathbf{B}}$$

where **B** represents the rule body, X, Y and Z are entity variables, and r is the head relation. This score regards facts that are not in KG as false, in another word, it implements a closed world setting.

Table 3 reports the results. We compare Ruleformer with other differentiable rule mining methods. The main difference between these methods and ours is that Ruleformer could parse the rule according to the environment, but others parse the same rules for each triplet with the same relation. There is a significant improvement in standard confidence with our model and algorithm. Specifically, the standard confidence on *UMLS* improves about 43.7% and on *FB15K-237*, the improvement is 25.5%. This is not only because the model generates suitable rules during training, but also because our parsing algorithm outputs the rules which can be mapped to an existing path in KG.

Methods	**UMLS**			**FB15K-237**		
	TOP			**TOP**		
	50	**100**	**200**	**50**	**200**	**500**
Neural-LP(T=2)	.228	.239	.221	.020	.044	.033
Neural-LP(T=2)	.104	.145	.153	.020	.031	.034
DRUM(T=2)	.400	.350	.303	.058	.036	.048
DRUM(T=3)	.340	.284	.202	.020	.039	.027
Ruleformer(T=2)	**.837**	**.793**	**.740**	.241	**.338**	**.310**
Ruleformer(T=3)	.680	.652	.573	**.313**	.322	.282

Table 3: Average confidence of top ranked rules on datasets *UMLS* and *FB15K-237*. The superscript [2] or [3] means with rule length 2 or 3, respectively.

4.5 Case Study

As we introduced, for different prediction tasks, we hope our model generate suitable rules in a better way with context of head entity into consideration rather than using them in the same order.

To verify the ability of generating appropriate rules of our model, we choose four test triplets from *WN18RR* that are shown in Figure 3.

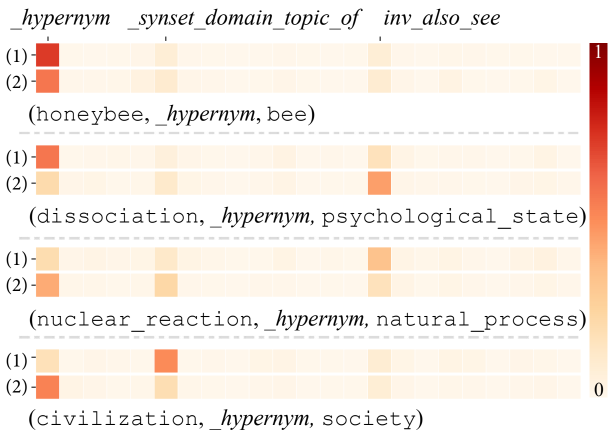

Figure 3: Decode output of four different triplets with the same head relation *_hypernym*. (1) and (2) means the first and second step.

Input these triplets into our model with rule length $T = 2$, and the output contains four sets of data which represent the probability of each relation for two steps in different cases. The heatmap in Figure 3 shows the output. Each row in the figure represents the probability for all relations in a step, and each pair of rows correspond to rules with different scores for the specific triplet which is showed below each subfigure. The darker the color is, the higher probability the relation has for current step. Note that we added inverse relations and *self-loop* so there are more than twice relations in the dataset. The relations with high probability are marked in the top of the figure.

As the figure shows, the four triplets have the same head relation `hypernym` which means a word that is more generic than a given word, while the head entities are different. for different triples with the same relation to be predicted, the probability of relations generated for reasoning in each step are not the same. Take the first case as an example, the rule with the highest confidence is '`hypernym(X, Z)` $\leftarrow$ `hypernym(X, Y)` $\wedge$ `hypernym(Y, Z)`'. While for the second example, the rule is '`hypernym(X, Z)` $\leftarrow$ `hypernym(X, Y)` $\wedge$ `inv_alsoSee(Y, Z)`' where `inv` means inverse relation. These different but all correct rules shows that our method successfully generate different rules and contextual information of triplets do have an impact on rule reasoning.

5 Conclusion

In this paper, we draw attention to using context to assist rule mining. We regard it as a sequence generation problem, and design a converter turning graph structure into a sequence. We propose a transformer-based model, Ruleformer, which utilizes the subgraph of head entity when learning rules over knowledge graph. The experiment results show that in a specific environment, our proposed model can mine and select different suitable rules. The performance on link prediction task and rule parsing also improves with our model and parsing algorithm. Future work may focus on a more effective way of rule mining and consider more complex forms of rules.

Acknowledgements

This work is funded by NSFCU19B2027/91846204.

References

Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. 2019. Tucker: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 5184–5193. Association for Computational Linguistics.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

Mingyang Chen, Wen Zhang, Yushan Zhu, Hongting Zhou, Zonggang Yuan, Changliang Xu, and Huajun Chen. 2022. Meta-knowledge transfer for inductive knowledge graph embedding. In *SIGIR*, pages 927–937. ACM.

Sanxing Chen, Xiaodong Liu, Jianfeng Gao, Jian Jiao, Ruofei Zhang, and Yangfeng Ji. 2021a. Hitter: Hierarchical transformers for knowledge graph embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 10395–10407. Association for Computational Linguistics.

Zhuo Chen, Jiaoyan Chen, Yuxia Geng, Jeff Z. Pan, Zonggang Yuan, and Huajun Chen. 2021b. Zero-shot visual question answering using knowledge graph. In *ISWC*, volume 12922 of *Lecture Notes in Computer Science*, pages 146–162. Springer.

William W Cohen. 2016. Tensorlog: A differentiable deductive database. *arXiv preprint arXiv:1605.06523*.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 132–141. Association for Computational Linguistics.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

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

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

Yuxia Geng, Jiaoyan Chen, Wen Zhang, Yajing Xu, Zhuo Chen, Jeff Z. Pan, Yufeng Huang, Feiyu Xiong, and Huajun Chen. 2022. Disentangled ontology embedding for zero-shot learning. In *KDD*, pages 443–453. ACM.

Qingyu Guo, Fuzhen Zhuang, Chuan Qin, Hengshu Zhu, Xing Xie, Hui Xiong, and Qing He. 2020. A survey on knowledge graph-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*.

Stanley Kok and Pedro M. Domingos. 2007. Statistical predicate invention. In *ICML*, volume 227 of *ACM International Conference Proceeding Series*, pages 433–440. ACM.

Daphne Koller, Nir Friedman, Sašo Džeroski, Charles Sutton, Andrew McCallum, Avi Pfeffer, Pieter Abbeel, Ming-Fai Wong, Chris Meek, Jennifer Neville, et al. 2007. *Introduction to statistical relational learning*. MIT press.

Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67.

Yankai Lin, Zhiyuan Liu, Huan-Bo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 705–714. The Association for Computational Linguistics.

Christian Meilicke, Melisachew Wudage Chekol, Daniel Ruffinelli, and Heiner Stuckenschmidt. 2019. Anytime bottom-up rule learning for knowledge graph completion. In *IJCAI*, pages 3137–3143. ijcai.org.

Stefano Ortona, Venkata Vamsikrishna Meduri, and Paolo Papotti. 2018. Robust discovery of positive and negative rules in knowledge bases. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1168–1179. IEEE.

Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. 2019. DRUM: end-to-end differentiable rule mining on knowledge graphs. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 15321–15331.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, and Jianfeng Gao. 2018. M-walk: Learning to walk over graphs using monte carlo tree search. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6787–6798.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *7th International Conference on Learning Representations*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, pages 1499–1509. The Association for Computational Linguistics.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Po-Wei Wang, Daria Stepanova, Csaba Domokos, and J Zico Kolter. 2019. Differentiable learning of numerical rules in knowledge graphs. In *International Conference on Learning Representations*.

Chi-Man Wong, Fan Feng, Wen Zhang, Chi-Man Vong, Hui Chen, Yichi Zhang, Peng He, Huan Chen, Kun Zhao, and Huajun Chen. 2021. Improving conversational recommender system by pretraining billion-scale knowledge graph. In *ICDE*, pages 2607–2612. IEEE.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations*.

Fan Yang, Zhilin Yang, and William W. Cohen. 2017. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 2319–2328.

Yuan Yang and Le Song. 2020. Learn to explain efficiently via neural logic inductive learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. QA-GNN: reasoning with language models and knowledge graphs for question answering. In *NAACL-HLT*, pages 535–546. Association for Computational Linguistics.

Wen Zhang, Bibek Paudel, Liang Wang, Jiaoyan Chen, Hai Zhu, Wei Zhang, Abraham Bernstein, and Huajun Chen. 2019a. Iteratively learning embeddings and rules for knowledge graph reasoning. In *WWW*, pages 2366–2377. ACM.

Wen Zhang, Bibek Paudel, Wei Zhang, Abraham Bernstein, and Huajun Chen. 2019b. Interaction embeddings for prediction and explanation in knowledge graphs. In *WSDM*, pages 96–104. ACM.

Wen Zhang, Chi Man Wong, Ganqiang Ye, Bo Wen, Wei Zhang, and Huajun Chen. 2021. Billion-scale pre-trained e-commerce product knowledge graph model. In *ICDE*, pages 2476–2487. IEEE.