Knowledge Inference and Knowledge Completion Methods using Neuro-Symbolic Inductive Rules

Won-Chul Shin
Computer Science Department
Soongsil University
Seoul, South Korea
ocshin1201@naver.com

Hyun-Kyu Park
Computer Science Department
Soongsil University
Seoul, South Korea
phkchr09@gmail.com

Young-Tack Park
Computer Science Department
Soongsil University
Seoul, South Korea
park@ssu.ac.kr

***Abstract*— In knowledge graph completion, a symbolic reasoning method establishes a human readable rule by analyzing an imperfect knowledge graph and infers knowledge omitted by an inference engine. However, the entire rules cannot be defined based on a large-scale knowledge graph. This study proposes a method, based on a knowledge graph, that can facilitate end-to-end learning and induce rules without several processing steps that require direct human involvement. The proposed method combines the concept of unification used in symbolic reasoning and deep learning for training vectors expressing symbols. It trains the vectors expressing relations of rule schemas defined to induce rules based on a given knowledge graph. Furthermore, the performance of the proposed method is evaluated against neural theorem prover and the greedy neural theorem prover, which are recently developed neuro-symbolic models, based on four benchmark datasets. The experimental results verify that the proposed method induces more significant rules in less training time. Furthermore, this study conducted an experiment on knowledge graph completion, implemented by an inference engine. Based on the experiment results, it was confirmed that the rules induced by the proposed model can indeed effectively complete missing knowledge.**

Keywords—knowledge graph completion, unification, rule schema, neuro-symbolic, neural theorem prover

I. INTRODUCTION

First, confirm that you have the correct template for your paper size. Of late, various studies have been conducted to develop methods for solving the problems on imperfection of knowledge graphs (KGs). Large-scale KGs such as FreeBase [1], YAGO [2], and DBpedia [3] provide an effective basis for many important AI tasks such as semantic search, link prediction [4], and question answering [5]. One such method is neural link prediction, which applies neural networks, facilitating an end-to-end learning process in which a learning system can receive input data and learn the preferred objects without performing necessary processing steps that require direct human intervention. This method also provides a function for powerful generalization related to data accuracy. When a model learns training data and uses types of data

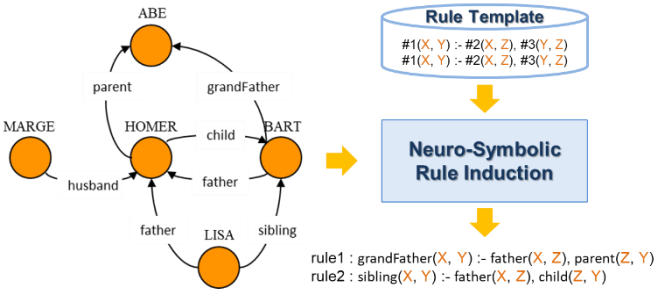

Fig. 1. Example of neuro symbolic rule induction

other than training data as input, then the accuracy of learning based on other types of data is similar to that based on the training data. However, there are limitations; for example, a large amount of training data is required to train a model, and the prediction result cannot be efficiently analyzed. A symbolic reasoning system requires a small amount of training data or sometimes even this is not necessary. It can also analyze a reasoning result. However, it has problems, such as the need for people to define rules by themselves and the inability to perform generalizations. A recent study [6] on completion of KGs proposed a neural theorem prover (NTP), which is a neuro-symbolic system combining neural network training with a symbolic reasoning system. It manages to retain the robustness of both systems while discarding their respective limitations. Therefore, it is estimated that this system exhibits a better performance than the existing systems. Furthermore, NTPs provide neural networks that facilitate end-to-end learning, similar to the neural link prediction method, and these neural networks are recursively established based on a backward chaining algorithm implemented by Prolog [7]. In particular, NTPs replace a homogeneity test between symbols for symbolic unification with that of similarity calculation related to vectors expressing symbols. Subsequently, they train query-based inferences and vectors expressing symbols. Neural networks

Algorithm 1 Comparison of symbolic unification and neural unification

1: **symbolic unification** $(\mathrm{H}, \mathrm{Q}, \mathrm{S} = (S_\psi, S_\rho))$
2: $S'_\psi = S_\psi \cup_i T_i$
with $T_i = \begin{cases} \{ H_i/Q_i \} & if\ H_i \in \mathcal{V} \\ \{ Q_i/H_i \} & if\ Q_i \in \mathcal{V}, H_i \notin \mathcal{V} \\ \emptyset & otherwise \end{cases}$
3: $S'_\rho = \begin{cases} True & if\ (H_i \notin \mathcal{V}) = (Q_i \notin \mathcal{V}) \\ False & if\ (H_i \notin \mathcal{V}) \neq (Q_i \notin \mathcal{V}) \end{cases}$
4: **return** (S'_ψ, S'_ρ)

1: **neural unification** $(\mathrm{H}, \mathrm{Q}, \mathrm{S} = (S_\psi, S_\rho))$
2: $S'_\psi = S_\psi \cup_i T_i$
with $T_i = \begin{cases} \{ H_i/Q_i \} & if\ H_i \in \mathcal{V} \\ \{ Q_i/H_i \} & if\ Q_i \in \mathcal{V}, H_i \notin \mathcal{V} \\ \emptyset & otherwise \end{cases}$
3: $S'_\rho = \min\{S_\rho\} \cup_{H_i, Q_i \notin \mathcal{V}} \{L2similarity(E_{H_i}, E_{Q_i})\}$
4: **return** (S'_ψ, S'_ρ)

of NTPs can be trained to induce logical rules that can be efficiently analyzed.

NTPs establish neural networks recursively through a backward chaining process to derive results of queries given based on rule templates, which define KGs and rule schemas. Presently, entire facts that exist in KGs are enumerated at every stage of the backward chaining process to calculate similarity between symbols instead of variables. In this method, the calculation complexity is $O(|\mathfrak{K}|n)$ when the number of sub queries for a KG $\mathfrak{K}$, is n. Owing to this, the calculation complexity increases significantly with a corresponding increase in the KG scale. The greedy neural theorem prover GNTP [8], a method developed to solve this problem, reduces the search space required for sub queries based on nearest neighbor search (NNS) [9]. This study proposes a neuro-symbolic rule induction model, which combines similarity calculation and symbolic matching in a backward chaining process to reduce the search space required for sub queries and thereby induce rules more effectively and efficiently than GNTPs.

Both the rule induction model proposed in this study and the NTP learn vectors expressing symbols in a training process using a backward chaining algorithm based on a KG and a rule template. A rule template consists of rule schemas (e.g. < #1(X, Y) :- #2(X, Z), #3(Z, Y) >) and has rule relations (e.g. #1, #2, #3) that can be trained. These models learn the vectors expressing rule relations in a training process and induce human readable rules through a certain decoding process. Fig. 1 shows an example of a process of inducing a neuro-symbolic rule based on a rule schema from a KG. The rule induced through this process can be applied to achieve knowledge completion through a symbolic-based inference performed by an inference engine. Furthermore, this study proposed a method that applies both the neuro-symbolic rule induction model and an integrated inference engine system to induce a rule from an incomplete KG and derive a complete KG based on the induced rule and the inference engine.

II. Related work & background

A. Studies on the Completion of KGs

Research on KGs has constantly faced challenges owing to the management of large-scale KGs and completion of missing information in them. Neural link prediction and symbolic reasoning based on rules are mainly used to automatically complete KGs. To perform neural link prediction, a KG is embedded in a vector space. Representative neural link prediction methods include TransE [10], TransR [11], DistMult [12], ConvE [13], ComplEx [14], and RotatE [15]. ConvE, which applies convolution neural networks (CNNs) [16] to KG embedding, exhibits excellent performance in automatic KG completion. However, these neural link prediction methods have limitations such as the requirement for a large amount of data to train a model and the inability to analyze the prediction results derived from the model. In contrast, symbolic reasoning performs query-based inference through a forward or backward chaining process reflecting logical rules defined by experts. Semantic web rule language (SWRL) [17] inference is a symbolic inference method that has been extensively examined to enhance inference engines for efficient inference. In addition, knowledge inference through inference engines in a spark [18] environment performs well in terms of inference time and computing resources. Through such research, these methods have been widely used to complete KGs. However, these symbolic reasoning methods involve overheads such as considerable expert time and a high cost to implement the process of expressing relations and providing rules for large-scale KGs. Moreover, relations and rules should be adjusted according to changes whenever knowledge is added or existing knowledge is adjusted.

B. Neural Theorem Prover

NTPs applying neural unification are operated based on vectors expressing symbols, and their neural networks facilitate end-to-end learning that can verify queries in a KG. These neural networks are recursively established, based on backward chaining inference implemented by Prolog. In particular, NTPs adopt a neural unification based on vectors expressing symbols and a function for calculating similarity between these vectors, to combine a symbolic reasoning process with a process of learning vectors expressing symbols. They also employ rule templates represented as relations (e.g. #1, #2, #3), which consist of embedding vectors that are unspecified and can be trained, and define these templates as parameterized rules [19] in terms of vector expression training. The parameterized rule (e.g. < #1(X, Y) :- #2(X, Z), #3(Z, Y) >) follows the basic form of rules used by Prolog. Based on ":-", its left and right sides refer to a conclusion and a premise, respectively. NTPs learn vectors expressing symbols to make vectors of relations of rule templates similar to those of certain relations of triples in KGs. Through this process, these provers can induce logical rules that can be analyzed.

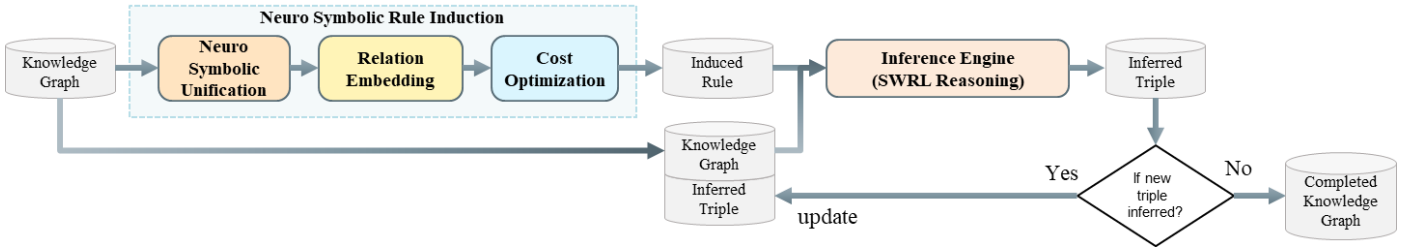

Fig. 2. Neuro-symbolic based knowldege completion system architecture

C. *Unification*

A unification calculation is used to perform fundamental symbolic- and query-based inference. In the unification process, S_ρ, which refers to a state of success of unification, and S_ψ, which is a substitution, are returned. $S_\rho$returns TRUE, if the symbols are analyzed to be homogeneous through a homogeneity test on these symbols instead of variables of two facts and FALSE if these symbols are not homogenous. If a corresponding variable to an entity is detected, a substitution S_ψ is created. However, a corresponding variable to a predicate is not allowed.

As the concept of unification does not exist in neural systems, the NTP adopted neural unification based on calculation of similarity between symbols. The neural unification process calculates similarity between embedding vectors of symbols instead of conducting a homogeneity test on symbols. Thus, S_ρ refers to the minimum similarity between comparable vectors. In this process, similarity between symbols with similar meanings (e.g. father and parent) can be considered through similarity calculation. However, neural unification requires embedding of entire symbols in a KG. Moreover, calculation of similarity between embedding vectors requires a considerable amount of time and cost compared to a homogeneity test on symbols.

Algorithm 1 describes the processes of symbolic unification and neural unification, where Q is a query such as grandfather(BART, ABE), H is a head (e.g. grandfather(X, Y)) of rules such as grandfather(X,Y):- father(X,Z) and parent(Z,Y)), and $\mathcal{V}$ is a variable of a rule such as (X, Y).

III. PROPOSED METHOD

A. *Systme Design*

Fig. 2 shows the architecture of the integrated system proposed in this study. First, this system operates a neuro-symbolic rule induction model to induce rules. Subsequently, it uses an inference engine to perform knowledge inference based on the induced rules and derive a complete KG through inference of missing knowledge.

B. *Proof Tree based on Neuro-symbolic Unification*

NTPs form proof trees by applying neural unification. In this method, the scale of a calculation graph increases geometrically with a corresponding increase in the scale of a KG. Therefore, the amount of calculation for a large-scale KG is limited. The neuro-symbolic unification method proposed in this study calculates the similarity of only the corresponding vectors to factor relations and carries out a homogeneity test on corresponding symbols to entities. Because of these characteristics, this method can effectively reduce the search space required for queries and forms proof trees to select only paths that can generate rules. Fig. 3 shows the process of generating a proof tree through neuro-symbolic unification. When a query triple Q is grandFather(BART, ABE), unification of Q and a head H (#1(X,Y)) of a rule is performed. Subsequently, a homogeneity test on entities is carried out. If a variable is detected, a substitution is generated. grandFather, a relation of a query triple, and #1, a relation of a rule template, are expressed as embedding vectors of $E_{grandFather}$ and $E_{\#1}$, respectively. The Euclidean similarity between the vectors is calculated to obtain a unification score S_ρ. When the entire relation sets of a KG and a rule template are R, S_ρ is calculated based on the following equation.

$$S_\rho = \exp(-\left\| E_{H_i} - E_{Q_i} \right\|_2) \quad \text{if } H_i, Q_i \in R \tag{1}$$

Subsequently, a substitution for a variable is generated, and BART and ABE are bound to a rule variable X,Y to obtain a substitution S_ψ, shown as {X/BART, Y/ABE}. Next, a substitution is applied to a rule #2(X,Z). X, the body of a rule, is replaced with BART in this process, and a sub-query shown as [#2, BART, Z] is generated. Unlike existing NTPs which perform unification of a sub-query and all the triples in a KG, the proposed method carries out symbolic matching to search a triple that is to be unified with a sub-query from a KG. This method refers to substitution values to search a triple to be unified with the sub-query. Here, BART, replaced by a substitution, is an entity corresponding to the subject of the sub-query. Consequently, father(BART, HOMER), a triple where the subject is BART, is searched from the KG and unified with the sub-query. Similarly, substitutions for the entire bodies of rule templates are obtained, and triples that can be used for unification are searched to generate a proof tree. As the last step for forming a proof tree, a proof score S' is calculated for each proof path generated in the proof tree. S' considers the mean of a unification score S_ρ, which is derived according to the number of sub-queries of rule heads and corresponding proof paths and is calculated based on the following equation.

$$S' = \text{avg}(\cup_i S_{\rho i}) \tag{2}$$

When the number of sub-queries is n for a KG $\mathfrak{K}$, the existing NTP shows a calculation complexity of $O(|\mathfrak{K}|n)$. However, this method shows a calculation complexity of

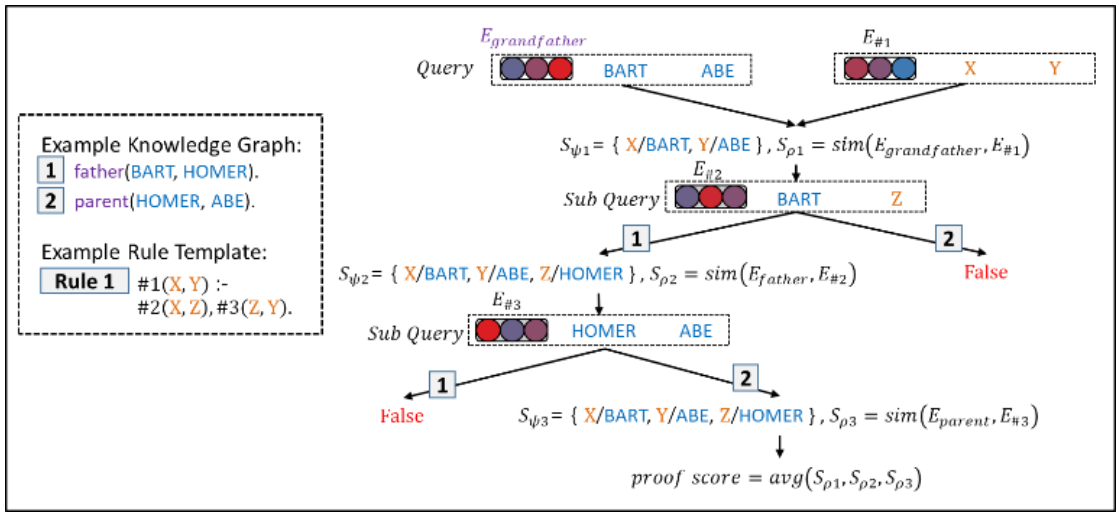

Fig. 3. Example of proof tree generation through neuro-symbolic unification

$O(|\mathfrak{K}'|n)$ because it performs unification based on only a fact $\mathfrak{K}'$, which satisfies a substitution through symbolic matching. If only a single fact corresponds to a substitution, the calculation complexity of this method decreases to $O(|1|n)$.

C. Relational Embedding Learning Method

A proof tree can be generated through a backward chaining process based on neuro-symbolic unification. To induce a rule, a model should perform a learning process to make vectors of relations of rule templates similar to those of relations of proof paths generated through neuro-symbolic unification. When the number of proof paths, which can be derived from a query triple(Q) and a rule template(RT) in the form of < #1(X, Y) :- #2(X, Z), #3(Z, Y) >, is n, RT is augmented to be applied to multiple proof paths. Augmentation of RT is conducted according to the rule number and the augment number. If the augment number is k for the 0th RT_0 < #1(X, Y) :- #2(X, Z), #3(Z, Y) >, an example of the augmented rule template is as follows.

Augmented RT_0
#1_0_0(X, Y) :- #2_0_0(X, Z), #3_0_0(Z, Y)

…

#1_0_k (X, Y) :- #2_0_k (X, Z), #3_0_k (Z, Y)

For example, #1_0_k refers to the first relation of the k^{th} augment of the 0th RT. The augment rule template generates a proof tree in the same manner as that indicated in Fig. 3. To facilitate embedding learning and cost optimization, a proof score is derived from n proof path(s), which is/are generated based on k augmented rule templates with regard to a query Q. A scoring function f used to derive this proof score is as follows.

$$f(Q)= \min_{n} (\max_{k}(U_{n,k} S^{'k}_{n})) \quad (3)$$

Here, the mean of similarity between relations of proof paths of proof score S', which is derived through the aforementioned process, should be 1. The mean of similarity between relations should also be 0 for the negative proof paths in which relations of relation sets of proof paths derived through unification are randomly sampled. To fulfill these conditions, cross-entropy loss is minimized based on the negative log-likelihood loss function indicated below.

$$L_{f(Q)}= \sum_{(Q\in \mathfrak{K}, y)} -y \log f(Q) -(1-y) \log (1-f(\tilde{Q})) \quad (4)$$

Each relation of a rule template trained through the process described above is decoded as a relation of the most similar triple to induce a rule. The rule induced through the aforementioned method can then be used to facilitate new knowledge inference based on triple information provided.

IV. Experiments

A. Dataset

This study used Nations, UMLS, Kinship [20], and FB122 [21], which were the bench mark data used in existing NTPs and GNTPs to verify the performance of the proposed rule induction method. Nations contains 14 relations, 55 entities, and 1793 true facts, UMLS contains 46 relations, 135 entities, and 5827 true facts and Kinship contains 26 relations, 104 entities, and 9618 true facts. The FB122 is a subset of FB15K-237, comprising 10,2881 triples, 9,738 entities, and 122 relations, as well as 47 rules that can be leveraged by models for link prediction tasks.

Furthermore, an experiment was conducted on the degree of increase in the amount of knowledge in a large-scale KG caused by knowledge inference, using Kdata and WiseKB. Both Kdata and WiseKB are larger in size than benchmark data. Kdata and WiseKB are large-scale KGs built in Korean, consisting mainly of triples with relations such as birthplace, genre, occupation, and nationality. Table 1 shows the statistics of KGs used in the experiment.

TABLE I. SUMMARY OF THE EXPERIMENT DATA

Dataset	#Train	#Test	#Entities	#Relations
Nations	1,592	201	55	14
UMLS	5,216	661	135	46
Kinship	8,544	1,074	104	26
FB122	91,638	11,243	9,738	122
Kdata	2,850 k		1,871 k	15,278
WiseKB	15,581 k		4,077 k	389

B. *Experimental Method*

To verify the performance of the proposed rule induction model, a comparison was performed with NTPs and GNTPs, which were developed in previous studies for rule induction and inference performance. Specifically, it compared the amount of time required for relation embedding based on four types of KGs (i.e. Nations, UMLS, Kinship, and FB122). The hardware specifications used in the learning time comparison experiment were an Intel i9-7900x CPU, NVIDIA GeForce GTX 1080 Ti GPU, and 128 GB RAM. The parameters used in all comparative experiments were conducted for 100 epochs, 3 rule templates: < #1(X, Y) :- #2(X, Z), #3(Z, Y) >, < #1(X, Y) :- #2(X, Y) >, and < #1(X, Y) :- #2(Y, X) > with an augment number of 20. The augment number 20 increases the number of learning parameters because it allows 20 rules to be induced in each of the three rule schemas < #1(X, Y) :- #2(X, Z), #3(Z, Y) >, < #1(X, Y) :- #2(X, Y) >, and < #1(X, Y) :- #2(Y, X) >. It also compared the inference result obtained based on the induced rule and the results of test sets. Additionally, with regard to Kdata and WiseKB, an experiment was performed on knowledge inference based on the induced rule by an inference engine to calculate the degree of increase in the amount of knowledge. With regard to Kdata and WiseKB, which had the highest number of triples, sub-KGs for inferring eight relations were generated. The eight selected relations would be the Test Triple relations that would be inferred through the rules. After selecting the target relation, the relation path to the target relation was extracted from the KG using the random walk algorithm of the path ranking algorithm PRA [22]. For example, if the relation path extracted from the target relation nationality is bornIn→nation, it can be converted into rules nationality(X,Y):-bornIn(X,Z), nation(Z,Y). In this manner, based on the rule set generated for all target relations, 10% of the triples corresponding to the target relation that were extracted from the KG were used as test sets, and the remaining triples were used as training sets. Subsequently, these datasets utilized 100,578 and 627,118 triples, respectively, to induce rules. Knowledge inference was implemented based on the induced rules, and training sets were implemented by the inference engine.

TABLE II. TRAINING TIME RESULT

Dataset	Avg #path/query	Training Time		
		NTP	GNTP	Our model
Nations	711	2 h 12 m	**5 m**	42 m
UMLS	45	10 h 13 m	1 h 18 m	**44 m**
Kinship	29	21 h 18 m	2 h 3 m	**23 m**
FB122	2	-	30 h 54 m	**2 h 19 m**

TABLE III. MODEL INFERENCE RESULT CONPARISON

Dataset	Induced Rules			Inferred Triples		
	NTP	GNTP	Our model	NTP	GNTP	Our model
Nations	21	28	**94**	52	84	**186**
UMLS	11	40	**65**	291	434	**520**
Kinship	12	37	**89**	490	628	**844**
FB122	-	27	**34**	-	4,889	**5,543**

V. EXPERIMENTAL RESULT

Table 2 shows the result of a comparative experiment on the amount of relation embedding training required for rule induction. The NTP required a significantly large amount of relation embedding training time, because it listed and scored the entire proof paths available for queries based on neural unification. This method unifies each query triple and sub-query with the entire triples that exist in a KG. For this reason, calculation complexity increases exponentially, as the scale of a KG increases. GNTP, a method proposed to solve such problems, performs fact selection based on NNS. Thus, it involved higher calculation cost than the proposed method based on symbolic matching in this study. For the Nations dataset, the number of triplets used for embedding learning is small, but because the KG concerns the relationship between countries in the world, the domain of the KG is limited to countries. Therefore, the number of proof paths that can be generated from one query triple can be higher than that of other benchmark data, and the proposed method considers all of the proof paths that can be generated by rules through symbolic matching on entities. As shown in Table 2, the average number of proof paths that the query triple can have in the nations data is 711, which is the largest number compared to other benchmark data used in the experiment. For this reason, the proposed method showed more time consumption compared to GNTP. However, as the number of triples increased, the proposed method required less time to derive rules than GNTP. In the case of FB122, it was impossible to derive rules due to a memory problem with

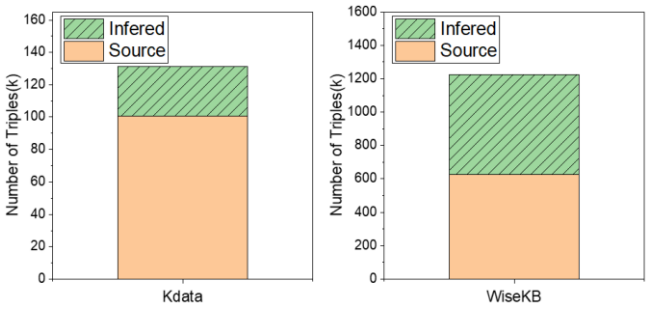

Fig. 4. Number of Inferred Triples

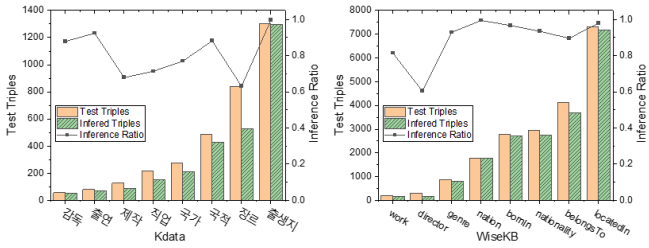

Fig. 5. Kdata and WiseKB Knowledge Inference

NTP, but compared to the size of the knowledge graph, the average number of proof paths that a query triple can have is 2, so it shows a learning speed more than 10 times faster than GNTP. In addition, the learning speed comparison experiment showed the largest speed improvement compared to the previous study. In the experiment on analyzing the performance of models for rule induction and triple inference, the number of rules induced through model training and the number of test triples that were inferred based on the rules were compared.

Table 3 shows the results of rule induction and triple inference. As indicated in this table, it was verified that the proposed method induced more rules that inferred test triples than the NTP and GNTP. It was also confirmed that the rules induced by the proposed model inferred more triples than the NTP and GNTP. FB122 comes with 47 logical rules that can be used in models along with data. The test data of FB122 is separated into test-1 (5,057 triple) and test-2 (6,186 triple) depending on whether the test set can be inferred through the provided logical rules. All the triples in test-2 can be inferred from the triplets in the training set through the provided logic rules, and we used this test-2 in the experiment. Through the experiment, we were able to derive 34 rules out of the 47 provided rules, and it was confirmed that 5543 triples, which is 89% of the 6,186 test triples, can be inferred from the derived rules.

Fig. 4 shows the induced rules based on Kdata and WiseKB and the number of triples inferred by the inference engine. As WiseKB comprised a higher number of triples than Kdata, the former also had a higher number of rule instances and inferred more instances than the latter. The inference result indicated that 30,751 triples were inferred from 100,578 triples based on Kdata. Thus, the degree of

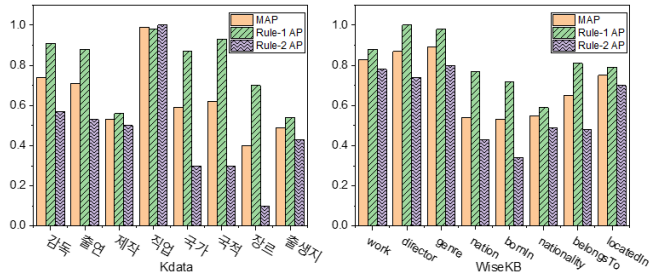

Fig. 6. Map results for Kdata and WiseKB induced rules

increase in the amount of knowledge based on a training set used in the experiment was calculated to be 30%. 596,350 triples were inferred from 627,118 triples based on WiseKB. Accordingly, the degree of increase in the amount of knowledge was calculated to be 95%.

Fig. 5 shows the experimental results for the number of triples of each eight target relations-specific test set and the number of test triples inferred from the inference engine, respectively, Kdata and WiseKB. The rules derived from the proposed rule induction model confirm that the missing knowledge was completed by inferring approximately 80% to 90% for each target relation.

Fig. 6 shows the results of mean average precision (MAP) for the two rule templates below with an augment number of 20 as a result of verification of the induced rules for Kdata and WiseKB.

Rule Templates

RT_0: #1(X, Y) :- #2(X, Z), #3(Z, Y)

RT_1: #1(X, Y) :- #2(X, Z), #3(Z, W) , #3(W, Y)

For Kdata and WiseKB, we generate rule sets, the correct set ofrules, from the relation path generated using PRA, so we can quantify how many of the rules were induced signifcantly. The induced rule was created by decoding the relations (e.g. #1_0_0, #2_0_0, #3_0_0) of each of the Augmnted Rule Templates into the relations of the KG with the highest similarity and using the minimum value of the Euclidean similarity between each relation as the confidence score. The rules obtained from each rule template were sorted in descending order based on the obtained confidence score. The average precision was obtained for each of the two rule templates used, and the mean was taken to calculate the MAP. Among the 40 rules derived from each of the two rule templates used, the higher the confidence value of the meaningful rules that can infer a test triple, the higher the AP and MAP. Although the MAP and AP values are low because some of the induction rules for target relations are not present in the rulesets generated from PRA results, there were many significant rules to infer test triple when reasoning was performed on the test set using inference engines.

VI. Training Details

We use ADAM [23] with an initial learning rate of 0.001 and a mini-batch size of 60 (Proof paths generated from 20

query triples and create two corrupted proof paths per proof path generated from one query triple) for optimization. We apply a ℓ_2 regularization of 0.0001 to all model parameters. All vector representations and rule parameters are initialized to values between 0 and 1. We train all models for 100 epochs. All models are implemented in Pytorch. We use following rule templates where the number in front of the rule template indicates how often a parameterized rule of the given structure will be instantiated.

Nations, UMLS, Kinship & FB122

20 #1(X, Y) :- #2(X, Y).

20 #1(X, Y) :- #2(Y, X).

20 #1(X, Y) :- #2(X, Z), #3(Z, Y).

Kdata & WiseKB

20 #1(X, Y) :- #2(X, Z), #3(Z, Y).

20 #1(X, Y) :- #2(X, Z), #3(Z, W), #4(W, Y).

VII. Conclusion

This study proposed an integrated system that uses a neuro-symbolic model to train embedding vectors of relations in a KG and infers new knowledge based on the rules induced. Furthermore, it performed a knowledge inference experiment to verify the performance of the proposed system, and it was verified that the proposed model performed vector training and rule induction based on both small- and large-scale KGs by applying neuro-symbolic unification and relation embedding training. Based on this method, the proposed model solved the problem of scalability in existing NTPs and induced more rules. Furthermore, it was confirmed that the rules induced by the proposed model can be effectively used to infer new knowledge and complete missing knowledge.

Acknowledgment

This work was supported by the Institute of Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. 2017-0-00162, Development of Human-care Robot Technology for Aging Society)

References

[1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," In Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1247–1250, https://doi.org/10.1145/1376616.1376746, 2008.

[2] F.M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," Proceedings of the 16th International Conference on World Wide Web, ACM, 2007.

[3] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, and C. Bizer, "DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia," *Semantic Web* 6(2):167–195, 2015.

[4] L. Yao, C. Mao, and Y. Luo, "KG-BERT: BERT for knowledge graph completion," arXiv preprint arXiv:1909.03193, 2019.

[5] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners," OpenAI blog, 2019.

[6] T. Rocktäschel, and S. Riedel, "End-to-end Differentiable Proving," *Advances in Neural Information Processing Systems* 30: *Annual conference on Neural Information Processing Systems*, Long Beach, CA, December, 2017.

[7] H. Gallaire, "Logic and Data Bases," *Symposium on Logic and Data Bases, Centre d'Études et de Recherches de Toulouse*. New York: Plemum Press, 1977.

[8] P. Minervini, M. Bošnjak, T. Rocktäschel, S. Riedel, and E. Grefenstette, "Differentiable Reasoning on Large Knowledge Bases and Natural Language," In Proceedings of the Thirty-Fourth AAAI Conference Artificial Intelligence, California, AAAI Press, 2020.

[9] J. Johnson, M. Douze, and H. Jegou, "Scale similarity search with GPUs," arXiv preprint arXiv:1702.08734, 2017.

[10] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating Embeddings for Modeling Multi-relational Data," *Advances in Neural Information Processing Systems* 2: 2787–2795, 2013.

[11] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning Entity and Relation Embeddings for Knowledge Graph Completion," In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, January 2015.

[12] B. Yang, W. Yih, X. He, J. Gao, and L. Deng, "Embedding Entities and Relations for Learning and Inference in Knowledge Bases," In Proceedings of ICLR, 2015.

[13] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2D Knowledge Graph Embeddings," In Proceedings of the Thirty-Second AAAI Conference Artificial Intelligence, New Orleans, LA, February, 2018.

[14] T. Trouillon, J. Welbl, S. Riedel, E. Gaussier, and G. Bouchard, "Complex Embeddings for Simple Link Prediction," In ICML, 48:2071–2080, 2016.

[15] Z. Sun, Z.-H. Deng, J.-Y. Nie, and J. Tang, "Rotate: Knowledge graph embedding by relational rotation in complex space," In ICLR, 2019.

[16] Y. Kim, "Convolutional neural networks for sentence classification," In EMNLP, pp. 1746–1751, ACL, 2014.

[17] I. Horrocks, P.F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean, "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," *Member Submission*, *W3c* 21(79):1–31, 2004.

[18] M. Zaharia, M.J. Chowdhury, M. Franklin, S. Shenker, and I. Stoica, "Spark: cluster computing with working sets," Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, 10:10–10, 2010.

[19] R. Socher, D. Chen, C.D. Manning, and A.Y. Ng, "Reasoning with Neural Tensor Networks for Knowledge Base Completion," In Proceedings of the 26th International Conference Neural Information Processing Systems, New York: Curran Associates Inc, 2013.

[20] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda, "Learning Systems of Concepts with an Infinite Relational Model," In AAAI, 381–388, 2006.

[21] S. Guo, Q. Wang, L. Wang, B. Wang, and L. Guo, "Jointly Embedding Knowledge Graphs and Logical Rules," In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, Texas: Association of Computational Linguistics, https://doi.org/10.18653/v1/D16-1019, 2016.

[22] M. Gardner and T. Mitchell "Efficient and Expressive Knowledge Base Completion Using Subgraph Feature Extraction," Proceedings of the 2015 Conference on Empirical Methods in Natural Language.

[23] P. Kingma Diederik and Ba. Jimmy "Adam: A method for stochastic optimization," In International Conference on Learning Representations (ICLR), 2015.