Received June 8, 2021, accepted August 29, 2021, date of publication August 31, 2021, date of current version September 9, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3109443

Analogical Reasoning With Deep Learning-Based Symbolic Processing

HIROSHI HONDA AND MASAFUMI HAGIWARA, (Senior Member, IEEE)
Faculty of Science and Technology, Keio University, Yokohama 223-8522, Japan

Corresponding author: Hiroshi Honda (honda@keio.jp)

This work was supported in part by the Keio University Kenkyu-no-Susume Scholarship, and in part by the Keio Leading-edge Laboratory of Science and Technology (KLL) Ph.D. Program Research Grant.

ABSTRACT The authors propose analogical reasoning systems based on first-order predicate logic using deep learning. The proposed systems consist of a model combining recursive neural networks and Word2Vec. When unknown data is input in this trained model, similar outputs to those of the trained data are obtained. Previous studies on symbolic inference using deep learning have focused on deductive and inductive inferences, and they require rule templates that are manually created and provided to the models in advance. However, it is not enough to infer unknown rules from a large amount of data on the Internet. Analogical reasoning is a mode of inference that attempts to solve a new case based on similarity. Thus, it enables inference of rules that were challenging to comprehend in previous studies. As a result of the experiments, unknown rules can be efficiently inferred from known rules contained in the knowledge bases. Using the proposed systems, the authors can extract unknown rules from five of the eight datasets consisting of three types of knowledge bases described in Prolog. The proposed method is the first case of analogical reasoning based on the first-order predicate logic using deep learning. Furthermore, the models used in the proposed systems have shown high robustness when using Word2Vec. For this reason, it is suggested that the practical applications of the proposed systems enable efficient inferences from a large amount of data that include noisy data on the Internet.

INDEX TERMS Analogical reasoning, deep learning, knowledge base, prolog, symbolic processing, Word2Vec.

I. INTRODUCTION

Symbolic processing using neural networks has been widely studied. In the 1990s, symbolic processing using multilayer neural networks [1], [2] was studied. However, owing to the limitations of the available hardware and training data at the time, as well as the learning ability of the multilayer neural networks, practical results could not be achieved.

In the 2000s, with the emergence of deep learning, the learning ability of neural networks improved significantly. As a result, studies on symbolic processing using neural networks have again gained attention. Recently, studies have been conducted to learn about the first-order predicate logic using deep learning and to make inferences using the learned results.

Previous studies on symbolic inference using deep learning focused on deductive and inductive inferences [3]–[10]. However, these studies required rule templates that were manually created and provided to the models in advance. For example, “#1(X,Y):–#2(X,Z),#3(Z,Y).”. In this case, only rules that match the rule templates can be found, thereby alluding that these studies are not enough to infer unknown rules from a large amount of data on the Internet. To efficiently formulate inference, deductive and inductive inferences as well as methods that have been utilized in conventional artificial intelligence (AI) research (e.g., analogical reasoning and hypothesis inference) are important. Therefore, in this research, we focus on analogical reasoning using deep learning. Analogical reasoning [11]–[13] is a mode of inference that attempts to solve a new case based on the similarity with another case. To date, there have been studies on analogical reasoning using neural networks [14]–[17], however, our research is the first to conduct analogical reasoning using first-order predicate logic.

One of the studies related to analogical reasoning using first-order predicate logic is transfer learning using

The associate editor coordinating the review of this manuscript and approving it for publication was Sunil Karamchandani.

a probabilistic logic model (PLM) [18], [19]. Transfer learning using PLM aims to generate a learning model for a completely different domain using a model trained for one domain. As transfer learning using PLM transfers all the rules contained in the trained model to generate a hypothesis for another domain, the number of hypotheses becomes enormous. Meanwhile, our approach aims to discover unknown rules using a trained model. By using similarity, analogical reasoning can limit the hypothesis search space to obtain new rules. Unlike transfer learning, analogical reasoning cannot obtain almost all the rules of a certain domain, but new rules can be efficiently discovered. Furthermore, analogical reasoning can discover not only new rules in another domain from a model trained in one domain, but also unknown rules within the same domain.

Our study makes the following contributions:

1) Unknown rules can be inferred from a knowledge base that contains known rules.
2) Even if the size of the knowledge base is large, unknown rules can be extracted efficiently.

Our proposed system can efficiently extract unknown rules from the knowledge base. Furthermore, the proposed method is the first case of analogical reasoning based on the first-order predicate logic using deep learning. Using knowledge bases described in Prolog, we evaluate our proposed systems.

We begin by reviewing related research work in section II. In section III, we describe analogical reasoning, which is the learning target. In section IV, we propose analogical reasoning with deep learning-based symbolic processing. Lastly, in Section V, we report the experimental results of the proposed systems.

II. RELATED WORK

A. ANALOGICAL REASONING

Studies on AI and cognitive science have been focused on making machines perform analogies as humans do [11]–[13], [20]–[24]. Initially, structure-mapping theory [11] based on the analogy of the relation between objects was proposed. Then, an inference engine that performed analogies based on structure-mapping theory [12] was developed. Furthermore, an inference engine using similarities between objects and similarities in problem solving [13] was developed.

Recently, studies have been conducted to learn about the knowledge graph using analogical reasoning with deep learning [25], [26]. However, it is difficult for them to treat the first-order predicate logic.

B. SYMBOLIC PROCESSING WITH NEURAL NETWORKS

Prior to the emergence of deep learning, studies on symbolic processing learning and inference using multilayer neural networks were conducted. Studies on inference for propositional logic [27]–[29] and first-order predicate logic [30]–[35] have been conducted. In addition, studies dealing with analogical reasoning [14], [17] were conducted.

After the emergence of deep learning, deductive and inductive inferences based on the first-order predicate logic [4], [7]–[10] were studied using graph neural networks. Later, studies using feedforward networks [5] and recurrent neural networks [3], [6] were also conducted. These studies required forms of rules or information of rules, to be provided to models in advance, which is not sufficient to infer unknown rules from a large amount of data on the Internet. However, although analogical reasoning has been studied for images and simple symbols [15], [16], it has not been considered for the first-order predicate logic.

C. TRANSFER LEARNING

One of the studies related to analogical reasoning is transfer learning. Since the 2000s, studies on transfer learning for symbolic knowledge representation [18], [19], [36], [37] have been conducted. In particular, research on transfer learning of predicate logic using the PLM [18], [19] was being performed. In transfer learning using PLM, based on a PLM of one domain, a PLM of another domain was created. Transfer learning using PLM has a problem in that the number of hypotheses increases considerably because a model of another domain was generated using all the rules included in the model. Furthermore, a study was being conducted to heuristically limit the range of hypothesis searches by hand [38]. Therefore, it is difficult to use transfer learning using PLM to discover new rules from a huge amount of information such as from the Internet.

D. ZERO-SHOT LEARNING

One of the studies related to analogical reasoning is zero-shot learning [39], [40]. Zero-shot learning aims to classify if unknown categories are included. It classifies data including unknown categories by measuring the similarity between input data and known categories. Some zero-shot learning studies used word embedding to measure similarity [41], [42], while others used human knowledge to measure similarity [43], [44]. However, although zero-shot learning can classify unknown categories, it is difficult to infer unknown rules from knowledge bases.

III. ANALOGICAL REASONING WITH SYMBOLIC PROCESSING

Here, we used Prolog [45], a subset of first-order predicate logic for symbolic processing. In this section, we describe the methods of analogical reasoning based on representations in Prolog.

A. MECHANISM OF ANALOGICAL REASONING

Fig. 1 shows the mechanism of analogical reasoning. The target domain is the domain for which the properties should be known, while the source domain is subjected to analogy. If there is similarity between problem A in the source domain and problem B in the target domain, solution A, the solution to problem A, is mapped to solution B, the solution to problem B. Thereafter, an analogy can be made by finding

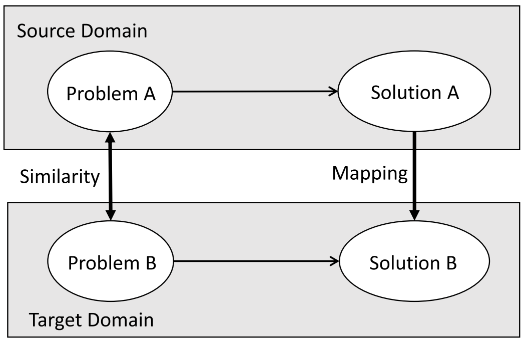

FIGURE 1. Mechanism of analogical reasoning.

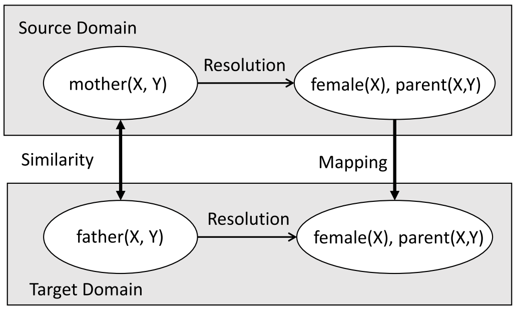

FIGURE 2. Example of analogical reasoning for rules.

similarities between the problem in the target domain and that in the source domain.

There are two types of similarity: (1) similarity between objects in the target and source domains, and (2) between relationships within those objects. Analogical reasoning for rules performs analogies based on the similarity of relationships, and analogical reasoning for unification performs analogies based on the similarity of objects. In our proposed systems, the reasoning ability is enhanced by combining both analogical reasoning for rules and unification. Our proposed systems use Word2Vec to measure similarities for both objects and relationships.

B. ANALOGICAL REASONING FOR RULES

We describe how to apply the mechanism of analogy to Prolog rules. Fig. 2 shows an example of analogical reasoning for rules. Consider the case where the rule shown in (1) exists in the source domain and the fact shown in (2) exists in the target domain. Rule (1) expresses that if X is a woman and X is Y's parent, then X is Y's mother. In Prolog programs, uppercase letters such as 'X' and 'Y' mean variables. The one in front of the operator ':-' is called the head, and the one in the back the body. The rule is that if the body is true, then the head is also true. The operator ',' indicates a conjunction of clauses. This means that the clauses must be true. Equation (2) expresses the fact that X is Y's father.

$$mother(X, Y) : -female(X), parent(X, Y). \tag{1}$$

$$father(X, Y). \tag{2}$$

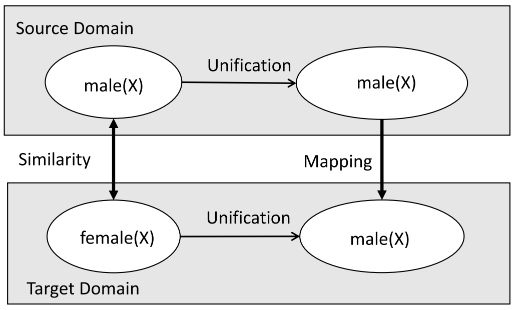

FIGURE 3. Example of analogical reasoning for unification.

Here, we consider the head on the left side of the rule in (1) as a problem and the body on the right side of the rule in (1) as a solution. Then, by finding the similarity between the head of (1) and the fact in (2), the body of (1) is mapped to the body of the rule with the head of (2). In other words, the rule shown in (3) can be inferred. This method of analogy is hereinafter called "analogical reasoning for rules".

$$father(X, Y) : female(X), parent(X, Y). \tag{3}$$

Now, referring to rule (3), the father is a woman, which is against the fact. Just by analogical reasoning for rules, it is not always possible to obtain the correct rules. Therefore, we use analogical reasoning for the unification described in the next section.

C. ANALOGICAL REASONING FOR UNIFICATION

We describe how to apply the analogy mechanism to unification, which is the determination of whether a given goal is the same as a fact [45]. In Prolog programs, we can ask whether two clauses are unified by connecting the two clauses with the operator '='. Fig. 3 shows an example of analogical reasoning for unification. Consider the case where the question shown in (4) is "true" in the target domain, and the fact shown in (5) exists in the source domain.

$$male(X) = male(X). \tag{4}$$

$$female(X). \tag{5}$$

Here, we consider the clause on the left side of (4) as a problem and that on the right side of (4) as a solution. Then, by finding similarities between the clause on the left side of (4) and the fact in (5), the clause on the right side of (4) is mapped to the solution of (5). As such, it can be inferred that the question shown in (6) becomes "true." This method of analogy is hereinafter called "analogical reasoning for unification".

By applying the unification shown in (6) to rule (3), as obtained by analogical reasoning for rules, the rule in (7) can be derived. In the rule shown in (7), the father is a male, which is the correct rule.

$$female(X) = male(X). \tag{6}$$

$$father(X, Y) : -male(X), parent(X, Y). \tag{7}$$

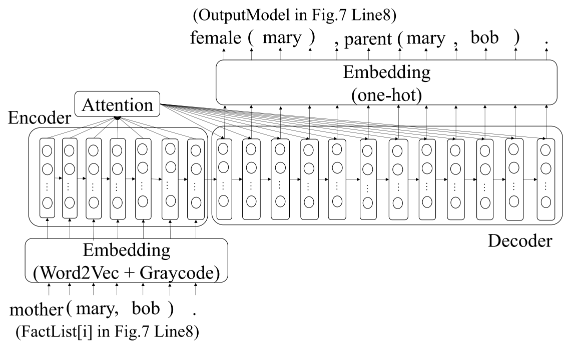
(OutputModel in Fig.7 Line8)
female (mary) , parent (mary , bob) .
Embedding (one-hot)
Attention
Encoder
Decoder
Embedding (Word2Vec + Graycode)
mother (mary, bob) .
(FactList[i] in Fig.7 Line8)

FIGURE 4. Proposed rule analogy model.

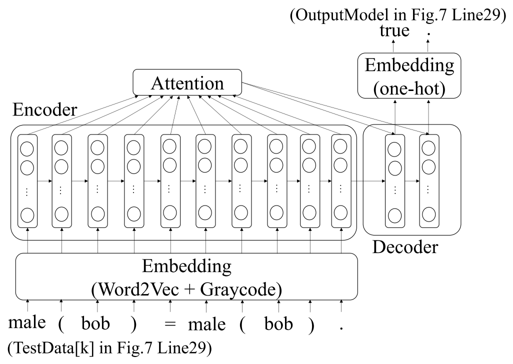
(OutputModel in Fig.7 Line29)
true .
Embedding (one-hot)
Attention
Encoder
Decoder
Embedding (Word2Vec + Graycode)
male (bob) = male (bob) .
(TestData[k] in Fig.7 Line29)

FIGURE 5. Proposed unification analogy model.

IV. PROPOSED ANALOGICAL REASONING WITH DEEP LEARNING-BASED SYMBOLIC PROCESSING

In this section, we first describe a method for implementing the analogical reasoning described in section III via deep learning. Next, we describe the proposed system that uses these methods for inferring rules from fact sets.

A. DEEP LEARNING-BASED SYMBOLIC PROCESSING

Using the model [6], which combines Seq2Seq with Attention [46] and Word2Vec [47]–[49], the learning of analogical reasoning for rules as well as unification is realized. The proposed model presents the advantage that resolution and unification can be performed even when unknown data are input using Word2Vec. We believe that a model using Word2Vec is effective in finding the similarity between problems in the source and target domains, which is the purpose of analogical reasoning. The similarity is measured by the distance between the embedding vectors of different words.

1) LEARNING RULE ANALOGY

Fig. 4 shows a rule analogy model, which is realized using the resolution model [6]. This model is trained to produce the word string of the body, which is on the right side of the rule when the word string of the head, which is on the left side of the rule, is input. Rules may include variables.

First, in the input embedding layer, the word sequence of the head is converted into a combined vector of word embedding by Word2Vec and Gray code [50]. Gray code has the property that presents the Hamming distance between adjacent codes, before and after, always as 1. On the one hand logical symbols such as '(', ',', ')', '=', and '.' and proper nouns of atoms such as "mary" and "bob" are embedded in Gray code. On the other hand, common nouns of atoms such as "mother" are embedded in Word2Vec.

Word2Vec is a method for obtaining distributed representations of words from a large amount of text data using a neural network. The input embedding layer uses GoogleNews-vectors-negative300 [47]–[49], a Word2Vec that has been trained. This has been trained on a dataset of approximately 100 billion words. Words are represented by 300-dimensional vectors, holding vectors of three million words.

Second, the combined vector generated by the input embedding layer is passed to Seq2Seq with Attention. Seq2Seq with Attention consists of the following: Encoder, Decoder, and Attention. When the Encoder receives an input sequence, it produces a compressed vector. Attention calculates the degree of attention given to each word in the input sequence based on the context of the output sequence. A weight that depends on the degree of attention is added to the compression vector. When the Decoder receives vectors from the Encoder and Attention, it generates an output sequence.

Long short-term memory (LSTM) [51] is used for the Encoder and Decoder. We apply Bi-LSTM, capable of handling future as well as past information to the Encoder. The Bi-LSTM consists of three layers and has a 128-dimensional output layer. A stateless LSTM that does not inherit short-term memory is applied to the Decoder. Stateless LSTM has a 128-dimensional output layer and uses Maxout [52] as the activation function.

Third, the output of Seq2Seq with Attention is passed to the output embedding layer, embedded in one-hot, and thereafter produced as the word string of the body.

For this model to learn, the rules of the source domain are given to the model as training data. For the learning in this study, the dropout rate was set to 0.1, the batch size was set to 128, and learning was performed for 20 epochs. Using Adam [53] as the optimizer, the parameters were set to $\alpha = 0.001$, $\beta1 = 0.9$, $\beta2 = 0.999$, and eps = 1e-08. By providing the facts of the target domain to the trained model, analogical reasoning for rules is realized.

2) LEARNING UNIFICATION ANALOGY

Fig. 5 shows a unification analogy model, which is realized by using the unification model [6]. This model is trained to insert a question as to whether it can be unified. It is likewise trained to produce "true" as the output when unification is possible and "false" when unification is not possible. The related questions may include variables. Unlike the prolog processing system, this model does not instantiate variables and outputs. For example, when the question shown in (8) is input to this model, "true." is the output instead

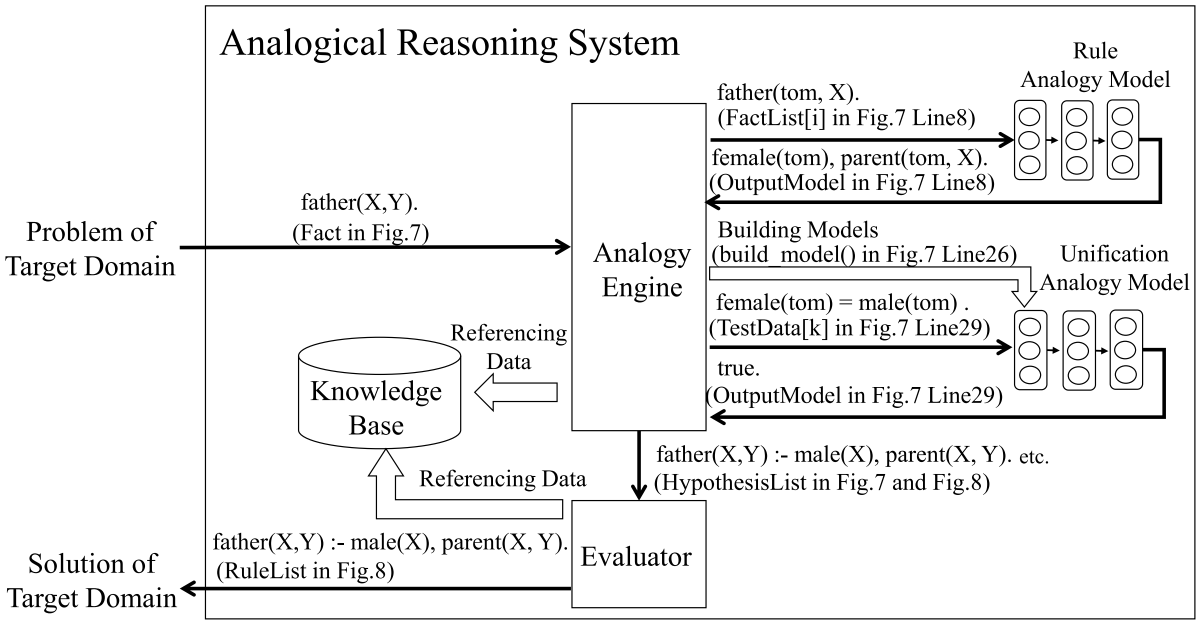
Analogical Reasoning System
Rule Analogy Model
father(tom, X).
(FactList[i] in Fig.7 Line8)
female(tom), parent(tom, X).
(OutputModel in Fig.7 Line8)
Problem of Target Domain
father(X,Y).
(Fact in Fig.7)
Analogy Engine
Building Models
(build_model() in Fig.7 Line26)
Unification Analogy Model
female(tom) = male(tom) .
(TestData[k] in Fig.7 Line29)
true.
(OutputModel in Fig.7 Line29)
Referencing Data
Knowledge Base
Referencing Data
father(X,Y) :- male(X), parent(X, Y). etc.
(HypothesisList in Fig.7 and Fig.8)
father(X,Y) :- male(X), parent(X, Y).
(RuleList in Fig.8)
Evaluator
Solution of Target Domain

FIGURE 6. **Proposed analogical reasoning system.**

of "X = bob.".

$$male(X) = male(bob). \tag{8}$$

First, in the input embedding layer, the word sequence of the question is converted into a combined vector of word embedding by Word2Vec and Gray code. Logical symbols such as '(', ',', ')', '=', and '.' and proper nouns of atoms such as "bob" are embedded in Gray code. Common nouns of atoms such as "male" are embedded in Word2Vec. GoogleNews-vectors-negative300 is applied to Word2Vec as well as to the rule analogy model.

Second, the combined vector generated by the input embedding layer is passed to Seq2Seq with Attention. The configuration of Seq2Seq with Attention is the same as that of the rule analogy model.

Third, the output of Seq2Seq with Attention is passed to the output embedding layer, embedded in one-hot, and thereafter produced as the word string of "true." or "false.".

To learn the proposed model, the question of whether it can be unified, which belongs to the source domain, is given to the model as training data. For the learning in this study, the dropout rate was set to 0.1, the batch size was set to 128, and learning was performed for 20 epochs. Using Adam as the optimizer, the parameters were set to $\alpha = 0.001$, $\beta 1 = 0.9$, $\beta 2 = 0.999$, and eps = 1e-08. By inputting questions based on facts, belonging to the source domain, into the trained model, analogical reasoning for unification is realized.

B. PROPOSED ANALOGICAL REASONING SYSTEM

This section describes the proposed analogical reasoning system using the rule analogy and the unification analogy models. When a certain fact is input into this system, an analogy is performed using the knowledge base described in Prolog, and rules, of which heads are the fact, are produced. For example, when the fact shown in (9) is input into this system, the rules shown in (10) are produced by analogical reasoning even if there is no rule with (9) in the knowledge base.

$$father(X, Y). \tag{9}$$

$$father(X, Y) : -male(X), parent(X, Y). \tag{10}$$

Fig. 6 shows the configuration of the proposed analogical reasoning system. This system consists of an analogy engine and an evaluator. When the analogy engine receives a fact, it generates multiple rules whose heads are the fact as hypotheses. Thereafter, it passes them to the evaluator. When the evaluator receives the hypotheses, it evaluates them using the knowledge base and produces verified hypotheses as outputs. The following describes the details of each component of this system.

1) ANALOGY ENGINE

The analogy engine performs analogies using the rule analogy model described in section IV-A-1, and the unification analogy model described in section IV-A-2. The analogy engine uses the rule analogy model, learned in advance, using the rules stored in the knowledge base. It uses the unification analogy model generated during the analogy process, using the facts stored in the knowledge base.

Fig. 7 shows the algorithm used by the analogy engine. First, the analogy engine references the knowledge base and lists facts that can be unified with Fact. (Lines 2–4) The facts are stored in *FactList*. The function *unify()* performs the same processing as the unification of the Prolog processing system. At this time, terms created by replacing arguments of the facts with a variable are added to *FactList*. (Line 5) Replacement with a variable by the function *variabilization()*

```
Algorithm 1:   Analogical Reasoning Algorithm
Input: a fact written in Prolog  Fact
Output: rules written in Prolog  HypothesisList
1:  FactList ← []
2:  for i = 0 to size(KnowledgeBase) do
3:      if unify(KnowledgeBase[i], Fact)
4:          append(FactList, KnowledgeBase[i])
5:          append(FactList, variableization(KnowledgeBase[i]))
6:  RuleResultList ← []
7:  for I = 0 to size(FactList) do
8:      OutputModel ← rule_analogy_model(FactList[i])
9:      append(RuleResultList, OutputModel)
10: RuleResults ← extract_high_correct_rate(RuleResultList , α)
11: HypothesisList ← []
12: for i = 0 to size(RuleResults) do
13:     AtomList ← []
14:     AtomList ← split_into_atoms(RuleResults[i])
15:     ConvertAtomList ← []
16:     for j = 0 to size(AtomList) do
17:         FactList1 ← []
18:         FactList2 ← []
19:         TrainingData ← []
20:         TestData ← []
21:         for k = 0 to size(KnowledgeBase) do
22:             if unify(KnowledgeBase[k], AtomList[j])
23:                 append(FactList1, KnowledgeBase[j])
24:             else
25:                 append(FactList2, KnowledgeBase[j])
26:         TrainingData ← convert_training_data(FactList2)
27:         TestData ← convert_test_data(FactList1, FactList2)
28:         unification_analogy_model ← build_model(TrainingData)
29:         UnificationResultList ← []
30:         for k = 0 to size(TestData) do
31:             OutputModel ← unification_analogy_model(TestData[k])
32:             append(UnificationResultList, OutputModel)
33:         UnificationResults ← extract_high_correct_rate(UnificationResultList, β)
34:         append(ConvertAtomList, UnificationResults)
35:     Hypothesis ← convert_atom(RuleResult, ConvertAtomList)
36:     append(HypothesisList, Hypothesis)
37: return HypothesisList
```

FIGURE 7. Analogical reasoning algorithm.

is an operation that replaces one of the fact arguments with a variable X. For example, if (11) is replaced by a variable, (12) and (13) are generated.

$$father(tom, bob). \tag{11}$$
$$father(X, bob). \tag{12}$$
$$father(tom, X). \tag{13}$$

Second, *FactList* is appended into the rule analogy model (the function *rule_analogy_model()*), and the analogy result is obtained (Lines 7–9). The results are stored in the *RuleResultList*. From *RuleResultList*, the top rules from the one with the highest ratio among the rules, of which the body arguments are all explained by the head argument, are extracted (Line 10). For example, if the head of the rule is (14), the arguments of (15), which is the body of the rule, are "tom" and "bob," like the head. Thus, all the arguments of the body are explained. Meanwhile, the arguments of (16), which is the body of the rule, are "jim" and "ken"; these are not explained by the arguments of (14). The rules are extracted by the function *extract_high_correct_rate()* and stored in *RuleResults*. The parameter α is the number of rules to be extracted.

$$father(tom, bob). \tag{14}$$
$$female(tom), parent(tom, bob). \tag{15}$$
$$female(tom), parent(jim, ken). \tag{16}$$

```
Algorithm 2:   Hypothesis Testing Algorithm
Input: rules written in Prolog  HypothesisList
Output : rules written in Prolog  RuleList
1:  RuleList ← []
2:  for i = 0 to size(HypothesisList) do
3:      UnifiedRuleList ← unify(KnowledgeBase, HypothesisList[i])
4:      Head ← extract_head(HypothesisList[i])
5:      UnifiedHeadList ← unify(KnowledgeBase, Head)
6:      if size(UnifiedRuleList) / size(UnifiedHeadList) >= γ
7:          append(HypothesisList, HypothesisList[i])
8:  return RuleList
```

FIGURE 8. Hypothesis testing algorithm.

Next, hypotheses are generated for each rule in *RuleResults* (Lines 12–36). The body of the rule is split into terms by the function *split_into_atoms()* (Line 14). For example, if there is a rule (17), the terms (18) and (19) are obtained. The terms are stored in *AtomList*.

$$father(X, Y) : -female(X), parent(X, Y). \tag{17}$$
$$female(X). \tag{18}$$
$$parent(X, Y). \tag{19}$$

The facts stored in the knowledge base are unified for each decomposed term (Lines 16–25). The facts that can be unified with *AtomList* are stored in *FactList1*, and those that cannot be unified are stored in *FactList2*. *FactList2* is used to create training data to learn a unification analogy model using the function *convert_trainning_data()* (Line 26). For example, when (20), a unifiable fact with the term (18) is obtained, the training data shown in (21) are created. The training data are stored in *TrainingData*.

$$male(tom). \tag{20}$$
$$male(X) = male(tom).true. \tag{21}$$

FactList1 and *FactList2* are combined to create test data for testing the unification analogy model using the function *convert_test_data()* (Line 27). For example, when (22), a fact that can be unified with the term (18), is obtained, the test data shown in (23) are created. The test data are sorted in TestData.

$$female(mary). \tag{22}$$
$$male(X) = female(mary).true. \tag{23}$$

A unification analogy model is built using *TrainingData* by the function *build_model()* (Line 28). After that, *TestData* is put into the unification analogy model (the function *unification_analogy_model()*), and the analogy result is obtained (Lines 30–32). The function *unification_analogy_model()* is trained from scratch in each iteration. This is because the atom to be analogized must be unknown to the model. For example, when test data containing the unknown word "female" are input to the trained unification analogy model, the model outputs results that can be unified with "female" by similarity expressed in Word2Vec. The results are stored in *UnificationResultList*. From *UnificationResultList*, the top terms from the one with the highest ratio of outputting "true"

TABLE 1. Training datasets of kinsources and correct answer rate of models.

	Rule Analogy Model			
Dataset	A-1	A-2		
Training Data	26,709	25,199		
Validation Data	3,215	3,196		
Test Data	3,325	3,804		
Correct answer Rate	0.8842	0.9008		
	Unification Analogy Model			
Dataset	B-1	B-2	B-3	B-4
Training Data	66,971	67,103	29,906	38,863
Validation Data	8,402	8,412	3,685	4,815
Test Data	8,363	8,383	3,705	4,877
Correct answer Rate	1.0000	1.0000	1.0000	1.0000

TABLE 2. Results of rule analogy models by kinsources.

Dataset	Number of Analogy Test Data	Form of Head	Form of Body	Match Rate (α=1)
A-1	1,215	mother(X,Y).	male(X), parent(X,Y).	0.6724
	1,180	daughter(X,Y).	male(X), parent(Y,X).	0.4712
	689	wife(X,Y).	male(X), spouse(X,Y).	0.3149
A-2	1,272	father(X,Y).	female(X), parent(X,Y).	0.5460
	1,341	son(X,Y).	female(X), parent(Y,X).	0.4459
	712	husband(X,Y).	female(X), parent(X,Y).	0.2725

are extracted (Line 35). The terms are extracted by the function *extract_high_correct_rate()* and stored in *ConvertAtomList*. The parameter β is the number of terms to be extracted.

Third, hypotheses are generated using *RuleResult* and *ConvertAtomList* by the function *convert_atom()* (Line 35). Specifically, a hypothesis is generated by replacing the term of the body of *RuleResult* with *ConvertAtomList*. For example, hypotheses (24), (25), (26), and (27) are obtained from *RuleResult* shown in (17). These hypotheses are stored in the *HypothesisList*. In addition, facts are input into this system one by one, and the hypotheses obtained by one process are meant for one clause.

$$father(X, Y) : -female(X), parent(X, Y). \tag{24}$$

$$father(X, Y) : -male(X), parent(X, Y). \tag{25}$$

$$father(X, Y) : -female(X), spouse(X, Y). \tag{26}$$

$$father(X, Y) : -male(X), spouse(X, Y). \tag{27}$$

2) EVALUATOR

When the evaluator receives *HypothesisList* from the analogy engine, hypothesis verification is conducted with reference to

TABLE 3. Results of unification analogy models by kinsources.

Dataset	Number of Analogy Test Data	Form of Question	Match Rate (β=3)
B-1	1,184	female(X)=male(X).	1.0000
B-2	1,184	male(X)=female(X).	1.0000
B-3	3,905	parent(X,Y)=spouse(X,Y).	1.0000
B-4	3,905	spouse(X,Y)=parent(X,Y).	1.0000

TABLE 4. Results of analogical reasoning system by kinsouces.

Dataset	Number of Hidden Rules	Discovered Rules	Discovery Rate
A-1	3	mother(X,Y):- female(X),parent(X,Y). daughter(X,Y):- female(X),parent(Y,X). wife(X,Y):- female(X),spouse(X,Y).	1.0000
A-2	3	father(X,Y):- male(X),parent(X,Y). son(X,Y):- male(X),parent(Y,X). husband(X,Y):- male(X),spouse(X,Y).	1.0000

the knowledge base. Fig. 8 shows the algorithm based on the evaluator test hypothesis.

First, the evaluator refers to the knowledge base to unify each hypothesis and collects rules that can be unified with the hypothesis by the function *unify()* (Line 3). The function *unify()* performs the same processing as the unification of the Prolog processing system. For example, if there is a hypothesis shown in (25), (28) is obtained as a rule that can be unified.

$$father(tom, bob) : -male(tom), spouse(tom, bob). \tag{28}$$

Second, the evaluator collects facts that can be unified with the head of the hypothesis (Lines 4–5) by the function *extract_head()*. For example, in the case of the hypothesis shown in (25), (29) is obtained as a fact that can be unified.

$$father(tom, bob). \tag{29}$$

Third, the evaluator calculates the ratio between the number of hypothetical rules that can be unified with the knowledge base and the number of facts that can be unified with the head of the hypothesis. If the ratio is greater than or equal to γ, the hypothesis is adopted. Otherwise, less than γ, it is rejected (Lines 6–7). Here, the parameter was set to $\gamma = 1.0$.

V. EVALUATION EXPERIMENTS

Using three types of knowledge bases described in Prolog, we trained models and built analogical reasoning systems. Next, the results of the evaluation experiments are shown

TABLE 5. **Training datasets of IMDb and correct answer rate of models.**

	Rule Analogy Model			
Dataset	A'-1	A'-2		
Training Data	90,904	77,103		
Validation Data	11,373	9,653		
Test Data	11,373	9,632		
Correct answer Rate	0.9489	0.9499		
	Unification Analogy Model			
Dataset	B'-1	B'-2	B'-3	B'-4
Training Data	88,868	91,229	90,821	46,650
Validation Data	11,147	11,404	11,361	5,815
Test Data	11,147	11,417	11,372	5,742
Correct answer Rate	1.0000	1.0000	1.0000	1.0000

TABLE 6. **Results of rule analogy models by IMDb.**

Dataset	Number of Analogy Test Data	Form of Head	Form of Body	Match Rate (α=1)
A'-1	2,076	production(X,Y).	person(X), movie(Y), produce(X,Y).	0.3801
A'-2	3,784	producer(X,Y).	company(X), movie(Y), produce(X,Y).	0.3554

TABLE 7. **Results of unification analogy models by IMDb.**

Dataset	Number of Analogy Test Data	Form of Question	Match Rate (β=3)
B'-1	243	person(X)=company(X).	0.8313
	280	person(X)=movie(X).	0.0000
B'-2	243	company(X)=person(X).	0.0000
	280	company(X)=movie(X).	0.0000
B'-3	78	movie(X)=person(X).	0.1538
	85	movie(X)=company(X).	0.0000
B'-4	1,176	produce(X,Y)=distribution(X,Y).	0.9932
	1,220	produce(X,Y)=sfx(X,Y).	0.0328
	1,654	produce(X,Y)=acting(X,Y).	0.0097

TABLE 8. **Results of analogical reasoning system by IMDb.**

Dataset	Number of Hidden Rules	Discovered Rules	Discovery Rate
A'-1	1	production(X,Y):- company(X),movie(Y), produce(X,Y).	1.0000
A'-2	1	-	0.0000

using three types of knowledge bases followed by their discussion.

A. EXPERIMENTS USING KINSOURCES

We used Kinsources [54] to train the models and build analogical reasoning systems. Kinsources is a collection of data representing blood relationships. The knowledge base described in Prolog consisted of 5,887 atoms and 10 kinds of predicates.

Table 1 shows the datasets used for the training rule analogy models and unification analogy models as well as the correct answer rates of the models. Datasets A-1 and A-2 were used to train rule analogy models and were built based on the rules included in the knowledge base. Datasets A-1 and A-2 consisted of three types of rules with predicates in the head. Datasets B-1, B-2, B-3, and B-4 were used to train the unification analogy models and were built based on the facts included in the knowledge base. Datasets B-1, B-2, B-3, and B-4 consisted of questions containing three types of predicates. The correct answer rate is the rate [6] at which the output of the model exactly matches the test data.

Table 2 shows the results of the rule analogy using the rule analogy models of Table 1. Table 3 shows the results of the unification analogy using the unification analogy models of Table 1. The match rate in Table 3 is the rate at which the output from the model is "true." Table 4 shows the results obtained by incorporating Datasets A-1 and A-2 into the analogical reasoning systems described in section IV-B. The parameters of the analogy engine were set to $\alpha = 1$ and $\beta = 3$. The number of hidden rules in Table 4 represents the number of rules removed from the dataset. The hidden rules were analogized by the analogical reasoning systems. The discovered rules in Table 4 were actually discovered by the proposed systems. The discovered rate in Table 4 was the rate of the rules discovered in the hidden rules. It can be said that the higher the discovery rate, the higher the ability of the systems to discover the rules.

B. EXPERIMENTS USING IMDb

We used IMDb, a movie- and television-related database [55], to train the models and build analogical reasoning systems. The knowledge base described in Prolog consisted of 20,658 atoms and 14 types of predicates.

Table 5 shows the datasets used in training the rule analogy and unification analogy models. Table 5 shows the accurate answer rates and was formulated based on the rules included in the knowledge base. Datasets A′-1 and A′-2 consisted of five types of rules with predicates in the head. Datasets B′-1, B′-2, B′-3, and B′-4 were used to train the unification analogy models and were built based on the facts included in the knowledge base. Datasets B′-1, B′-2, B′-3, and B′-4 consisted of questions containing eight types of predicates.

Table 6 shows the results of the rule analogy using the rule analogy models of Table 5. Table 7 shows the results of the

TABLE 9. Training datasets of clutrr and correct answer rate of models.

	Rule Analogy Model	
Dataset	A"-1	A"-2
Training Data	90,904	77,103
Validation Data	11,373	9,653
Test Data	11,373	9,632
Correct answer Rate	0.9489	0.9499
	Unification Analogy Model	
Dataset	B"-1	B"-2
Training Data	88,868	91,229
Validation Data	11,147	11,404
Test Data	11,147	11,417
Correct answer Rate	1.0000	1.0000

unification analogy using the unification analogy models of Table 5. Table 8 shows the results obtained by incorporating Datasets A′-1 and A′-2 into the analogical reasoning systems described in section IV-B. The parameters of the analogy engine were set to $\alpha = 1$ and $\beta = 3$. The number of hidden rules, the discovered rules, and the discovery rate in Table 8 have the same meaning as Table 4.

C. EXPERIMENTS USING CLUTRR

We used clutrr [56], a benchmark dataset generator used to test relational reasoning on text, to train models and build analogical reasoning systems. The dataset used here was generated with task1 and relation length 2. We built the knowledge base described in Prolog from the dataset. Atoms that appeared only in bodies were replaced with the variable Y. The knowledge base consisted of 42 atoms and 16 types of predicates.

Table 9 shows the datasets used in training the rule analogy and unification analogy models. It also shows the correct answer rates, built based on the rules included in the knowledge base. Datasets A″-1 and A″-2 consisted of seven types of rules with predicates in the head. Datasets B″-1 and B″-2 were used to train the unification analogy models and were built based on the facts included in the knowledge base. Both B″-1 and B″-2 consisted of questions containing seven types of predicates.

Table 10 shows the results of the rule analogy using the rule analogy models of Table 9. Table 11 shows the results of the unification analogy using the unification analogy models of Table 9. Table 12 shows the results obtained by incorporating Datasets A″-1 and A″-2 into the analogical reasoning systems described in section IV-B. The parameters of the analogy engine were set to $\alpha = 3$ and $\beta = 3$. The number of hidden rules, the discovered rules, and the discovery rate in Table 12 have the same meaning as Table 4.

TABLE 10. Results of rule analogy models by clutrr.

Dataset	Number of Analogy Test Data	Form of Head	Form of Body	Match Rate (α=3)
A"-1	415	mother(X,Z).	sister(Y,Z),father(X,Y).	0.1229
			son(Y,Z), grandfather(X,Y).	0.0964
			brother(Y,Z), father(X,Y).	0.0892
	569	sister(X,Z).	daughter(Y,Z), uncle(X,Y).	0.1406
			father(Y,Z),son(X,Y).	0.1002
			son(Y,Z),uncle(X,Y).	0.0545
	348	daughter(X,Z).	wife(Y,Z),son(X,Y).	0.1236
			son(Y,Z),brother(X,Y).	0.1178
			daughter(Y,Z), brother(X,Y).	0.0891
	360	grandmother(X,Z).	brother(Y,Z), grandfather(X,Y).	0.1667
			sister(Y,Z), grandfather(X,Y).	0.1417
			father(Y,Z),father(X,Y).	0.0361
	376	granddaughter(X,Z).	wife(Y,Z), grandson(X,Y).	0.1729
			husband(Y,Z), grandson(X,Y).	0.1649
	279	aunt(X,Z).	father(Y,Z),son(X,Y).	0.1111
			father(Y,Z), brother(X,Y).	0.0609
			father(Y,Z),father(X,Y).	0.0502
	249	niece(X,Z).	brother(Y,Z),son(X,Y).	0.1245
			daughter(Y,Z), brother(X,Y).	0.0683
			son(Y,Z),brother(X,Y).	0.0402
A"-2	430	father(X,Z).	father(Y,Z), mother(X,Y).	0.0651
			son(Y,Z), grandmother(X,Y).	0.0628
			father(Y,Z),sister(X,Y).	0.0419
	603	brother(X,Z).	brother(Y,Z), sister(X,Y).	0.1244
			father(Y,Z), daughter(X,Y).	0.0896
			sister(Y,Z),sister(X,Y).	0.0564
	329	son(X,Z).	son(Y,Z),sister(X,Y).	0.2158
			daughter(Y,Z), sister(X,Y).	0.0669
	338	grandfather(X,Z).	father(Y,Z), mother(X,Y).	0.2041
			father(Y,Z),sister(X,Y).	0.0740
			mother(Y,Z), mother(X,Y).	0.0266
	338	grandson(X,Z).	grandson(Y,Z), sister(X,Y).	0.1940
			husband(Y,Z), granddaughter(X,Y).	0.1511
	246	uncle(X,Z).	father(Y,Z),sister(X,Y).	0.1870
			father(Y,Z), mother(X,Y).	0.1138
			father(Y,Z), daughter(X,Y).	0.0285
	256	nephew(X,Z).	son(Y,Z),sister(X,Y).	0.3672

D. COMPUTATIONAL COMLPEXITY AND SEARCH EFFICIENCY COMPARISON

We compared the computational complexity and search efficiency of the baseline and proposed methods. The number of hypotheses generated is used as the computational complexity, while the number of rules discovered per number of hypotheses is used as the search efficiency.

In the baseline, templates such as (30) were created from known rules, and unknown rules were searched by setting atoms to anonymous variables of the templates by brute force. In (30), "_" means anonymous variables.

$$\_(X,Y) : -\_(X),\_\,(X,Y). \qquad (30)$$

TABLE 11. Results of unification analogy models by clutrr.

Dataset	Number of Analogy Test Data	Form of Question	Match Rate (β =3)
B"-1	15	father(X,Y)=mother(X,Y).	1.0000
	17	father(X,Y)=grandmother(X,Y).	0.1765
	13	brother(X,Y)=grandmother(X,Y).	0.1538
	21	son(X,Y)=daughter(X,Y).	0.5263
	12	son(X,Y)=granddaughter(X,Y).	0.4828
	20	grandfather(X,Y)=grandmother(X,Y).	0.2000
	19	grandson(X,Y)=daughter(X,Y).	0.5263
	26	grandson(X,Y)=granddaughter(X,Y).	0.5000
	18	uncle(X,Y)=grandmother(X,Y).	0.6667
	15	uncle(X,Y)=aunt(X,Y).	0.2000
	23	husband(X,Y)=daughter(X,Y).	0.4783
	14	husband(X,Y)=wife(X,Y).	0.4286
	21	husband(X,Y)=granddaughter(X,Y).	0.2381
B"-2	19	mother(X,Y)=son(X,Y).	0.9474
	15	mother(X,Y)=father(X,Y).	0.2667
	14	sister(X,Y)=grandson(X,Y).	0.5455
	13	sister(X,Y)=son(X,Y).	0.3333
	16	sister(X,Y)=father(X,Y).	0.1667
	21	daughter(X,Y)=son(X,Y).	1.0000
	19	daughter(X,Y)=grandson(X,Y).	0.0526
	20	grandmother(X,Y)=son(X,Y).	0.9474
	17	grandmother(X,Y)=father(X,Y).	0.2667
	26	granddaughter(X,Y)=grandson(X,Y).	0.3077
	12	granddaughter(X,Y)=son(X,Y).	0.0833
	25	aunt(X,Y)=son(X,Y).	1.0000
	16	aunt(X,Y)=father(X,Y).	0.6250
	11	wife(X,Y)=son(X,Y).	0.5455
	12	wife(X,Y)=grandson(X,Y).	0.3333
	12	wife(X,Y)=father(X,Y).	0.1667

TABLE 12. Results of analogical reasoning system by clutrr.

Dataset	Number of Hidden Rules	Discovered Rules	Discovery Rate
A"-1	28	mother(X,Z):- daughter(Y,Z),grandmother(X,Y).	0.5714
		mother(X,Z):- son(Y,Z),grandmother(X,Y).	
		mother(X,Z):- brother(Y,Z),mother(X,Y).	
		mother(X,Z):- sister(Y,Z),mother(X,Y).	
		sister(X,Z):- daughter(Y,Z),aunt(X,Y).	
		sister(X,Z):- son(Y,Z),aunt(X,Y).	
		sister(X,Z):- father(Y,Z),daughter(X,Y).	
		sister(X,Z):- mother(Y,Z),daughter(X,Y).	
		daughter(X,Z):- wife(Y,Z),daughter(X,Y).	
		grandmother(X,Z):- sister(Y,Z),grandmother(X,Y).	
		grandmother(X,Z):- mother(Y,Z),mother(X,Y).	
		grandmother(X,Z):- father(Y,Z),mother(X,Y).	
		grandmother(X,Z):- brother(Y,Z),grandmother(X,Y).	
		granddaughter(X,Z):- wife(Y,Z),granddaughter(X,Y).	
		granddaughter(X,Z):- husband(Y,Z),granddaughter(X,Y).	
		neice(X,Z):- brother(Y,Z),daughter(X,Y).	
A"-2	28	brother(X,Z):- father(Y,Z),son(X,Y).	0.2143
		brother(X,Z):- mother(Y,Z),son(X,Y).	
		grandfather(X,Z):- father(Y,Z),father(X,Y).	
		grandfather(X,Z):- mother(Y,Z),father(X,Y).	
		grandson(X,Z):- husband(Y,Z),grandson(X,Y).	
		grandson(X,Z):- wife(Y,Z),grandson(X,Y).	

The number of hypotheses generated at baseline was calculated using (31). R1 is the number of templates generated from the knowledge base, T is the number of terms contained in the template, and k is the number of terms in the knowledge base that have the same number of arguments as the template term.

$$n_b = \sum_{i=1}^{R_1} \prod_{j=1}^{T_i} k_{ij} \tag{31}$$

The number of hypotheses generated by the proposed method was calculated using (32). R2 is the number of facts input to this system, α (1 or 3 in our experiments), and β (3 in our experiments) are parameters of the analogy engine, and T (2 or 3 in our experiments) is the number of terms included in the rules of analogy results.

$$n_p = \sum_{i=1}^{R_2} \alpha_i \cdot \beta_i^{T_i} \tag{32}$$

Table 13 shows the results of comparing the baseline with the proposed method for computational complexity and search efficiency.

E. DISCUSSION

In the experiments using Kinsources, we extracted all three rules in each of the two datasets. Specifically, we inferred the mother from the father rules, the daughter from the son rules, and the wife from the husband rules. Inversely, we were able to infer the father from the mother rules, the son from the daughter rules, and the husband from the wife rules.

In the experiments using IMDb, we extracted rules from one dataset. However, we were unable to extract rules from another dataset. Specifically, we inferred the rule of producer from the rule of production. However, we were unable to infer the rule of production from the rule of producer. This might be because the unification of "company (X) = person (X)" could not be analogized.

In the experiments using clutrr, we extracted 22 rules from both datasets. The results of this experiment indicated that the proposed system could analogize even if the dataset contained multiple rules with the same head. (For example, "mother(X,Z):-daughter(Y,Z),grandmother(X,Y)." and "mother (X,Z):-son(Y,Z),grandmother(X,Y).") The proposed system could not discover all rules, but it discovered them at a rate of 57.14% for A'''-1, and 21.43% for A''-2. It is also possible to analogize repeatedly with this system using the discovered rules.

As a result of comparing the baseline and the proposed method, the number of hypotheses generated in the proposed method was smaller in any of the datasets than in the baseline. In addition, the search efficiency of the proposed method was

TABLE 13. Results of computational complexity and search efficiency.

Knowledge base	Method	Dataset	Number of hypotheses generated	Search efficienc y
Kinsources	Baseline	A-1	256	0.0117
		A-2	256	0.0117
	Proposed method	A-1	12	0.2500
		A-2	12	0.2500
IMDb	Baseline	A'-1	1089	0.0009
		A'-2	1089	0.0009
	Proposed method	A'-1	8	0.1250
		A'-2	4	N/A
clutrr	Baseline	A"-1	4096	0.0068
		A"-2	4096	0.0068
	Proposed method	A"-1	162	0.0988
		A"-2	183	0.0328

better than that of the baseline, except for the datasets for which the rules could not be found. It is considered that the larger the size of the knowledge base, the more advantageous the computational complexity and search efficiency of the proposed method, compared to the baseline.

Using the proposed analogical reasoning system, we extracted unknown rules from six datasets consisting of three types of knowledge bases. However, rules could not be extracted from all three datasets. The success or failure of rule extraction depends on how well the rule analogy and the unification analogy are obtained. In other words, if the similarity between the problem in the target domain and the problem in the source domain can be well represented by the models, it will be possible to extract many rules. The proposed models express the similarity of the problems using distributed word expression via Word2Vec. Therefore, there is a possibility that the performance of analogical reasoning systems can be improved by refining Word2Vec.

VI. CONCLUSION AND FUTURE WORK

Here, we proposed analogical reasoning systems based on first-order predicate logic using deep learning. We have implemented analogical reasoning by training symbolic processing models using deep learning and using these models which produce concepts similar to unknown data [6]. Experimental results show that our systems could efficiently extract unknown rules from previously learned rules through analogical reasoning.

Our system is the first to enable analogization by first-order predicate logic using deep learning. The proposed system enables the inference of rules that were difficult in previous studies. Furthermore, the models used in this system have demonstrated high robustness when using Word2Vec. For this reason, it is inferred that it is possible to make analogies from data that include noisy data on the Internet.

Future tasks include improving the performance of analogical reasoning systems by improving Word2Vec and analogical reasoning using large-scale data on the Internet.

REFERENCES

[1] G. E. Hinton, "Preface to the special issue on connectionist symbol processing," *Artif. Intell.*, vol. 46, nos. 1–2, pp. 1–4, Nov. 1990.

[2] D. S. Touretzky, "BoltzCONS: Dynamic symbol structures in a connectionist network," *Artif. Intell.*, vol. 46, nos. 1–2, pp. 5–46, Nov. 1990.

[3] N. Cingillioglu and A. Russo, "DeepLogic: Towards end-to-end differentiable logical reasoning," 2018, *arXiv:1805.07433*. [Online]. Available: http://arxiv.org/abs/1805.07433

[4] W. W. Cohen, "TensorLog: A differentiable deductive database," 2016, *arXiv:1605.06523*. [Online]. Available: http://arxiv.org/abs/1605.06523

[5] H. Dong, J. Mao, T. Lin, C. Wang, L. Li, and D. Zhou, "Neural logic machines," in *Proc. Int. Conf. Learn. Represent.*, New Orleans, LA, USA, 2019, pp. 1–22.

[6] H. Honda and M. Hagiwara, "Question answering systems with deep learning-based symbolic processing," *IEEE Access*, vol. 7, pp. 152368–152378, 2019, doi: 10.1109/ACCESS.2019.2948081.

[7] P. Minervini, M. Bosnjak, T. Rocktäschel, and S. Riedel, "Towards neural theorem proving at scale," 2018, *arXiv:1807.08204*. [Online]. Available: http://arxiv.org/abs/1807.08204

[8] T. Rocktaschel and S. Riedel, "End-to-end differentiable proving," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 3788–3800.

[9] L. Serani and A. S. D. Garcez, "Logic tensor networks: Deep learning and logical reasoning from data and knowledge," in *Proc. 11th Int. Workshop Neural-Symbolic Learn. Reasoning (NeSy) Co-Located With Joint Multi-Conf. Hum.-Level Artif. Intell. (HLAI)*, New York, NY, USA, 2016, pp. 1–12.

[10] G. Sourek, V. Aschenbrenner, F. Zelezny, and O. Kuzelka, "Lifted relational neural networks," in *Proc. NIPS Workshop Cogn. Comput., Integrating Neural Symbolic Approaches Co-Located With 29th Annu. Conf. Neural Inf. Process. Syst. (NIPS)*, Montreal, QC, Canada, 2015, pp. 1–21.

[11] D. Gentner, "Structure-mapping: A theoretical framework for analogy," *Cognit. Sci.*, vol. 7, no. 2, pp. 155–170, 1983.

[12] B. Falkenhainer, K. D. Forbus, and D. Gentner, "The structure-mapping engine: Algorithm and examples," *Artif. Intell.*, vol. 41, no. 1, pp. 1–63, Nov. 1989.

[13] K. J. Holyoak and P. Thagard, "Analogical mapping by constraint satisfaction," *Cognit. Sci.*, vol. 13, no. 3, pp. 295–355, Jul. 1989.

[14] M. Hagiwara and Y. Anzai, "Analogical reasoning by neural network," in *Proc. Seattle Int. Joint Conf. Neural Netw. (IJCNN)*, Seattle, WA, USA, 1991, p. 995.

[15] F. Hill, A. Santoro, D. Barrett, A. Morcos, and T. Lillicrap, "Learning to make analogies by contrasting abstract relational structure," in *Proc. Int. Conf. Learn. Represent.*, New Orleans, LA, USA, 2019, pp. 1–18.

[16] S. Reed, Y. Zhang, Y. Zhang, and H. Lee, "Deep visual analogy-making," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 1252–1260.

[17] Y. Salu, "A neural network for analogical reasoning," in *Proc. IEEE Int. Conf. Neural Netw. (ICNN)*, Orlando, FL, USA, Jun./Jul. 1994, pp. 4772–4777.

[18] R. Kumaraswamy, P. Odom, K. Kersting, D. Leake, and S. Natarajan, "Transfer learning via relational type matching," in *Proc. IEEE Int. Conf. Data Mining*, Atlantic City, NJ, USA, Nov. 2015, pp. 811–816.

[19] L. Mihalkova and R. Mooney, "Transfer learning by mapping with minimal target data," in *Proc. AAAI Workshop Transf. Learn. Complex Tasks*, Chicago, IL, USA, 2008, pp. 1–6.

[20] J. Carbonell and G. Jaime, "Learning by analogy: Formulating and generalizing plans from past experience," in *Machine Learning*, vol. 1. Berlin, Germany: Springer, 1983, pp. 137–161.

[21] D. Gentner and K. D. Forbus, "Computational models of analogy," *Wiley Interdiscipl. Rev., Cognit. Sci.*, vol. 2, no. 3, pp. 266–276, May 2011.

[22] D. Hofstadter, *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*. New York, NY, USA: Basic Books, 1996, p. 528.

[23] J. E. Hummel and K. J. Holyoak, "Distributed representations of structure: A theory of analogical access and mapping," *Psychol. Rev.*, vol. 104, no. 3, p. 427, 1997.

[24] L. B. Larkey and B. C. Love, "CAB: Connectionist analogy builder," *Cognit. Sci.*, vol. 27, no. 5, pp. 781–794, Sep. 2003.

[25] H. Liu, W. Yuexin, and Y. Yiming, "Analogical inference for multi-relational embeddings," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2168–2178.

[26] Y. Wu, X. Liu, Y. Feng, Z. Wang, R. Yan, and D. Zhao, "Relation-aware entity alignment for heterogeneous knowledge graphs," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 5278–5284.

[27] A. S. D. Garcez and G. Zaverucha, "The connectionist inductive learning and logic programming system," *Appl. Intell.*, vol. 11, no. 1, pp. 59–77, Jul. 1999, doi: 10.1023/A:1008328630915.

[28] J. W. Shavlik and G. G. Towell, "An approach to combining explanation-based and neural learning algorithms," *Connection Sci.*, vol. 1, no. 3, pp. 231–253, 1989.

[29] G. G. Towell and J. W. Shavlik, "Knowledge-based artificial neural networks," *Artif. Intell.*, vol. 70, nos. 1–2, pp. 119–165, 1994, doi: 10.1016/0004-3702(94)90105-8.

[30] L. Ding, "Neural prolog—The concepts, construction and mechanism," in *Proc. IEEE Int. Conf. Syst., Man Cybern. Intell. Syst. 21st Century*, Oct. 1995, pp. 3603–3608.

[31] M. V. M. Franca, G. Zaverucha, and A. S. D. Garcez, "Fast relational learning using bottom clause propositionalization with artificial neural networks," *Mach. Learn.*, vol. 94, no. 1, pp. 81–104, Jan. 2014, doi: 10.1007/s10994-013-5392-1.

[32] S. Holldobler, "A structured connectionist unification algorithm," in *Proc. 8th Nat. Conf. Artif. Intell.*, Boston, MA, USA, 1990, pp. 587–593.

[33] E. Komendantskaya, "Unification neural networks: Unification by error-correction learning," *Log. J. IGPL*, vol. 19, no. 6, pp. 821–847, Dec. 2011, doi: 10.1093/jigpal/jzq012.

[34] L. Shastri, "Neurally motivated constraints on the working memory capacity of a production systemfor parallel processing," in *Proc. 14th Annu. Conf. Cogn. Sci. Soc.*, Bloomington, IN, USA, 1992, pp. 159–164.

[35] P. Minervini, S. Riedel, P. Stenetorp, E. Grefenstette, and T. Rocktäschel, "Learning reasoning strategies in end-to-end differentiable proving," in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 6938–6949.

[36] W. Huayan and Q. Yang, "Transfer learning by structural analogy," in *Proc. 25th AAAI Conf. Artif. Intell.*, San Francisco, CA, USA, 2011, pp. 1–6.

[37] H. Thomas and K. Forbus, "Transfer learning through analogy in games," *Ai Mag.*, vol. 32, no. 1, pp. 70–83, 2011.

[38] R. Kumaraswamy, N. Ramanan, P. Odom, and S. Natarajan, "Interactive transfer learning in relational domains," *Künstliche Intelligenz*, vol. 34, no. 2, pp. 181–192, Jun. 2020.

[39] H. Larochelle, D. Erhan, and Y. Bengio, "Zero-data learning of new tasks," in *Proc. 23rd AAAI Conf. Artif. Intell.*, Chicago, IL, USA, 2008, pp. 1–6.

[40] M. Palatucci, D. Pomerleau, G. Hinton, and T. M. Mitchell, "Zero-shot learning with semantic output codes," in *Proc. NIPS*, Vancouver, BC, Canada, 2009, pp. 1–9.

[41] R. Socher, M. Ganjoo, C. D. Manning, and A. ng, "Zero-shot learning through cross-modal transfer," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 935–943.

[42] S. Reed, Z. Akata, H. Lee, and B. Schiele, "Learning deep representations of fine-grained visual descriptions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 49–58.

[43] Z. Xie, W. Cao, X. Wang, Z. Ming, J. Zhang, and J. Zhang, "A biologically inspired feature enhancement framework for zero-shot learning," in *Proc. 7th IEEE Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)/6th IEEE Int. Conf. Edge Comput. Scalable Cloud (EdgeCom)*, Aug. 2020, pp. 120–125.

[44] Z. Xie, W. Cao, and Z. Ming, "A further study on biologically inspired feature enhancement in zero-shot learning," *Int. J. Mach. Learn. Cybern.*, vol. 12, no. 1, pp. 257–269, Jan. 2021.

[45] I. Bratko, *Prolog Programming for Artificial Intelligence*, 2nd ed. Reading, MA, USA: Addison-Wesley, 1990, p. 597.

[46] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. ICLR*, San Diego, CA, USA, 2015, pp. 1–15.

[47] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proc. Workshop ICLR*, Scottsdale, AZ, USA, 2013, pp. 1–12.

[48] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. NIPS*, Lake Tahoe, NV, USA, 2013, pp. 3111–3119.

[49] T. Mikolov, W. Yih, and G. Zweig, "Linguistic regularities in continuous space word representations," in *Proc. NAACL HLT*, Atlanta, GA, USA, 2013, pp. 746–751.

[50] F. Gray, "Pulse code communication," U.S. Patent 263 205 8A, Mar. 17, 1953.

[51] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[52] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proc. 30th Int. Conf. Mach. Learn.*, Atlanta, GA, USA, 2013, pp. 1319–1327.

[53] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

[54] *Kinsources: A Collaborative Web Platform for Kinship Data Sharing*. Accessed: May 19, 2018. [Online]. Available: https://www.kinsources.net/

[55] *IMDb: Internet Movie Database*. Accessed: Nov. 29, 2019. [Online]. Available: http://klog.dinfo.unifi.it/data/imdb/ext.pl.gz

[56] K. Sinha, S. Sodhani, J. Dong, J. Pineau, and W. L. Hamilton, "CLUTRR: A diagnostic benchmark for inductive reasoning from text," 2019, *arXiv:1908.06177*. [Online]. Available: http://arxiv.org/abs/1908.06177