IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 18, 2018, accepted December 31, 2018, date of publication January 11, 2019, date of current version February 8, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2892140

Embedding Logic Rules Into Recurrent Neural Networks

BINGFENG CHEN[1], ZHIFENG HAO[1,2], XIAOFENG CAI[1], RUICHU CAI[1], WEN WEN[1], JIAN ZHU[1], AND GUANGQIANG XIE[1]

[1]Department of Computer Science, Guangdong University of Technology, Guangzhou 510006, China
[2]School of Mathematics and Big Data, Foshan University, Foshan 528000, China

Corresponding author: Zhifeng Hao (zfhaomail@gmail.com)

This work was supported in part by the Natural Science Foundation of China (NSFC)-Guangdong Joint Fund under Grant U1501254, in part by the NSFC under Grant 61876043 and Grant 61472089, in part by the Natural Science Foundation of Guangdong under Grant 2014A030306004 and Grant 2014A030308008, in part by the Guangdong High-level Personnel of Special Support Program under Grant 2015TQ01X140, in part by the Guangdong Provincial Key Laboratory of Cyber-Physical System under Grant 2016B030301008, and in part by the Pearl River S&T Nova Program of Guangzhou under Grant 201610010101.

ABSTRACT Incorporating prior knowledge into recurrent neural network (RNN) is of great importance for many natural language processing tasks. However, most of the prior knowledge is in the form of structured knowledge and is difficult to be exploited in the existing RNN framework. By extracting the logic rules from the structured knowledge and embedding the extracted logic rule into the RNN, this paper proposes an effective framework to incorporate the prior information in the RNN models. First, we demonstrate that commonly used prior knowledge could be decomposed into a set of logic rules, including the knowledge graph, social graph, and syntactic dependence. Second, we present a technique to embed a set of logic rules into the RNN by the way of feedback masks. Finally, we apply the proposed approach to the sentiment classification and named entity recognition task. The extensive experimental results verify the effectiveness of the embedding approach. The encouraging results suggest that the proposed approach has the potential for applications in other NLP tasks.

INDEX TERMS RNN, logic rules, sentiment classification, named entity recognition.

I. INTRODUCTION

Recurrent neural network (RNN) works well as sequence models because they are able to represent natural language sequences, and have been successfully applied to a variety of NLP tasks, including speech recognition [1], machine translation [2], sentiment classification [3], and named entity recognition [4].

In spite of the successes of RNN language models, the high complexity of the training procedure and a large number of training samples are the main problems of existing RNN models. Besides proposing a new training algorithm [5], incorporating the prior knowledge into the design and training of the network [6] is another promising and important research topic. However, most of the current research only processes simple prior knowledge, such as the dependency relation [7], and only addresses prior knowledge in the output layers.

According to the process of human cognition, people learn not only from simple rules but also from different forms of general knowledge graphs and rich experiences [8]. As shown in FIGURE 1, existing structured knowledge usually consists of a structured knowledge base [9], [10], social graph [11], and syntactic dependency [7]. A knowledge graph is commonly described as a head entity-relation-tail entity, such as ''carrot-prevent-cancer'' and ''carrot-contain-renieratene'', which can be further described by a probability graph. A social graph is another type of knowledge graph that can be used to express the relationship between people; for example ''Tom-friend of-Mary'', which means that Tom is a friend of Mary. Syntactic dependency rules are known as ''REL(R, L)-like advmod (quickly, very)'', can also be described by a probability graph.

Thus, how to incorporate the large-scale structured prior knowledge into the RNN is an important but still difficult problem. Fortunately, we find that a variety of knowledge graphs can be presented by a set of logic rules, which provides a unified formal presentation for the various structured prior knowledge. By extracting the logic rules from the structured knowledge and embedding the extracted logic rules into the RNN, we propose a general and concise way to incorporate prior knowledge in the RNN framework. The primary ideas of

2169-3536 © 2019 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

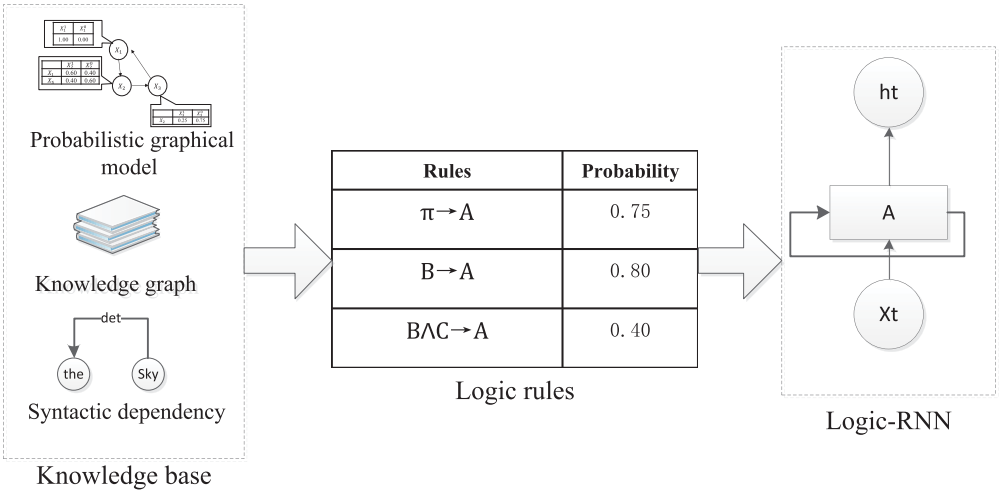


FIGURE 1. The framework of embedding logic rules into recurrent neural networks: 1) structured knowledge, e.g., probabilistic graphical model, knowledge graph, and syntactic dependency, are represented using logic rules; 2) the logic rules are embedded into the RNN.

our approach are presented in FIGURE 1. First, the structured knowledge graphs were presented by a set of logic rules. Second, the logic rules were embedded into the RNN structure by eliminating redundant feedback. Third, we applied the proposed approach to sentiment classification and named entity recognition. The experimental results showed that the logic rules embedded RNN model outperformed the RNN proposed by Mikolov *et al.* [12] and Hochreiter and Schmidhuber [13].

Our contributions in this paper are summarized as follows.

- We proposed a method to present a variety of structured knowledge graphs in the form of logic rules.
- We proposed a general and concise approach to embed the logic rules into RNN.
- We verified the effectiveness of the embedding approach on two typical NLP tasks.

The remainder of the paper is organized as follows: a summary of related work is presented in Section 2; in Section 3, we describe our method on embedding logic rules into RNN; the experimental results and discussions are presented in Section 4, and Section 5 concludes this paper.

II. RELATED WORK

In this section, we briefly introduce related work in two areas: first, we discuss the existing recurrent neural networks for sentiment classification and named entity recognition; second, we discuss previous research that combines rich structured knowledge with neural networks.

A. SENTIMENT CLASSIFICATION AND NAMED ENTITY RECOGNITION WITH RNN

There have been many recurrent neural networks proposed for sentiment classification and named entity recognition. Within sentiment classification, Tai *et al.* [14] proposed a tree-structured LSTM that outperforms other sequential models in sentiment classification but is limited to the tree structure. Wang et al. [15] used LSTM to generate a sentence representation for predicting the polarities of tweets. Tang et al. [3] utilized LSTM and the gated RNN on the document-level sentiment classification task. Additionally, Xu *et al.* [16] proposed a CLSTM that capture local and global emotional information. For named entity recognition, LSTM has been studied for the NER task by Hammerton [17]. Huang *et al.* [18] used a BLSTM for POS-tagging, chunking, and an NER task with heavy feature engineering. Furthermore, Chiu and Nichols [4] proposed a hybrid of the BLSTM and CNN to model both character-level and word-level representations. Lample *et al.* (2016) proposed a BLSTM-CRF model for NER, which utilized the BLSTM to model both the character- and word-level information, and used the same data pre-processing method as Chiu and Nichols [4]. Lastly, Ma and Hovy [19] used a CNN to model character-level information, and achieved better NER performance without the use of data pre-processing. Structured prior knowledge can help the training of RNN, but the previous models are unable to fully utilize them.

B. NEURAL NETWORKS WITH STRUCTURED KNOWLEDGE

The combination of logic information with human intuition and neural networks is beneficial in improving the training of neural networks. ENL-ANN [20], KBANN [21], and C-ILP [22], which are limited to propositional formulae, suggest that combing given rules with neural network structures lead to explanatory and knowledge acquisition. Rocktäschel *et al.* [23], [24] regularized entity-tuple and relation embedding via first-order logic rules. Hu *et al.* [5] used an iterative rule distillation process to effectively transfer rich structured knowledge, expressed in a declarative first-order logic language, into parameters of general neural networks. The results were encouraging on both the sentiment classification task and the named entity recognition task.

Our proposed approach is distinct in that we extracted logic rules from the structured knowledge base, and embedded these logic rules in the input layer of RNN. We show that the proposed approach is generic and can be used in a variety of applications.

III. THE PROPOSED METHOD

In this section, we present our approach of embedding the logic rules into the RNN. This was achieved by describing structured knowledge in the form of logic rules and modifying the existing RNN by embedding the logic rules. The framework of the proposed method is given in FIGURE 1. In this framework, the logic rules provide a unified presentation of the various prior knowledge, and further facilitate incorporating the prior knowledge into RNN without considering the various structure of the knowledge.

A. LOGIC RULES-BASED PRESENTATION OF STRUCTURED KNOWLEDGE

In this section, we show how to use the first order logic rule to describe the structured knowledge from the probabilistic graph, knowledge graph and syntactic dependency respectively. Here, we focus on the first order logic rules, because the first order logic rules are generic enough to present the knowledge from the various sources. The rules used in this work are listed as follows. Here $\wedge$, $\vee$, $\rightarrow$ and $\neg$ are logical operators, $\wedge$ is for conjunction, $\vee$ is for disjunction, $\rightarrow$ is for implication, $\neg$ is for negation.

- Rule 1: $\pi \rightarrow A$, if π, then A (π is a virtual variable that is always true for simplicity).
- Rule 2: $\pi \rightarrow \neg A$, if π, then $\neg$A.
- Rule 3: $B \rightarrow A$, if B, then A.
- Rule 4: $B \wedge C \rightarrow A$, if B and C, then A.
- Rule 5: $B \vee C \rightarrow A$, if B or C, then A.

Following, we will propose an abstract framework to extract the logic rules from the prior knowledge. Then, we will demonstrate the details of the abstract framework with the help of three toy examples, the probabilistic graphical model, knowledge graph, and syntactic dependency.

Firstly, the structured prior knowledge, e.g., probabilistic graphical model, knowledge graph and syntactic dependency, are all treated as a weighted graph $K = (\chi, \mathcal{E})$, with $\chi = \{X_1, \cdots, X_n\}$ is the node set, and $\mathcal{E} = \{< X_i \rightarrow X_j, \varepsilon, p_{ij} > | i, j \in \{1, 2, \ldots, n\}\}$ is the edge set. Here ε is type of the relation, and $p_{ij} \in [0, 1]$ is the relative weight (normalized to the range [0, 1]) of the relations.

Secondly, for each node X_i of the weighted graph K, the following two different steps are conducted to extract the corresponding logic rules. If the node X_i has no parental nodes, then the Rule 1 or Rule 2 are extracted, based on the configuration of the node A. Otherwise, Rule 3, Rule 4 or Rule 5 are extracted based on the number of parents node and the relationships among the nodes. Each extracted logic rule r_k is associated with the relation type ε_k and the relative weight p_k, in the form $< r_k, \varepsilon_k, p_k >$.

Thirdly, compound relations can be further generated as needed, by conducting composition operation on the rules generated in the previous step. And the relative weight of the generated rules is simply the multiplication of that of each rule, under the assumption that the rules are independent of each other.

By conducting the above abstract framework on the structured prior knowledge, a set of logic rules can be extracted, with each rule in the form $< r_k, \varepsilon_k, p_k >$. Following we will demonstrate the details on three toy examples.

1) PROBABILISTIC GRAPHICAL MODEL

Probabilistic graphical model use a graph-based representation for encoding a distribution over multi-variables [25]. And the graph is a factorized representation of a set of independences that hold in the specific distribution. The probabilistic graphical model is usually used as the encoding the prior knowledge in various areas, such as the medical science [26], social science [27].

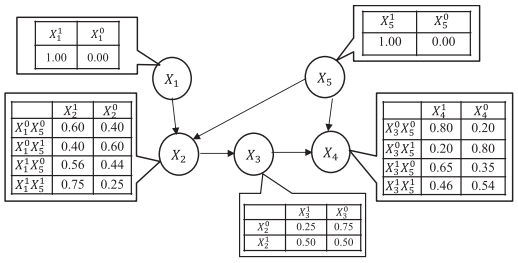


FIGURE 2. Toy Example of probabilistic graphical model. In the figure, each node is associated with a conditional probability table, which shows its corresponding probability conditioned on its parental node.

Figure 2 shows a typical probabilistic graphical model, as stated in the abstract framework, the nodes are connected by the directed edges, and there are conditional probability tables associated with the nodes. In this example, by considering the node X_1 and X_2, following typical structures can be extracted, they are $\pi \rightarrow X_1^0$ with weight 0.0, and $\pi \rightarrow X_1^1$ with weight 1.0, $X_1^0 \wedge X_5^0 \rightarrow X_2^0$ with weight 0.4, $X_1^0 \wedge X_5^0 \rightarrow X_2^1$ with weight 0.6, and so on. In this example, each cell of the conditional probability table is presented using one logic rules. Here, the relation types are all the same for all the edges, and the default value is set for all ε.

2) KNOWLEDGE GRAPH

Knowledge graph is a structured semantic knowledge base, which is used to describe the concept of the physical world and their interrelations in symbolic form. It is composed of entities as nodes and relations as different types of edges. The basic unit is the 'head entity (h)'-'relation (r)'-'tail entity (t)', which can be denoted as a triplet $< h, r, t >$. Freebase [9], WordNet [10] and other knowledge graphs have played important roles in various NLP tasks like Q&A and search engines.

In the toy example given in FIGURE 3, we can extract the following rules for the target node *cancer*, $DDP \rightarrow cancer$

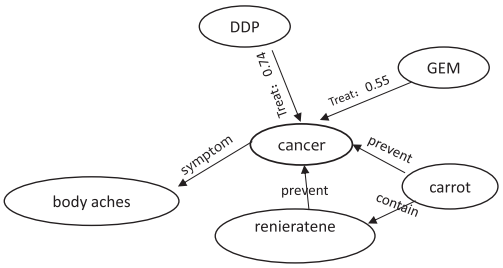


FIGURE 3. Toy example of knowledge graph. In the figure, each node is an entity, and some edges are associated with the type and the relative weight of relations.

with type 'treat' and weight 0.74, $GEM \rightarrow cancer$ with type 'treat' and weight 0.55 , $carrot \rightarrow cancer$ with type 'prevent' and weight 1.0, and $renieratene \rightarrow cancer$ with type 'prevent' and weight 1.0. Because there is no associated weight for the $carrot \rightarrow cancer$, the corresponding weight simply is set to 1. The difference between the probabilistic graphical model and the knowledge graph is that the probabilistic graphical model needs to enumerate all possible states of the parents, but the knowledge graph only focuses on the specified pairwise relations. Note that we also can generate more rules by conducting composition operators on the rules.

3) SYNTACTIC DEPENDENCY

Syntactic dependency describes the interdependence among the words of a sentence and reflects the syntactic collocation among the words that are associated with semantics. A typical syntactic dependency can be denoted as $REL(R, L)$, which is used to express the relationship between the R word and the L word.

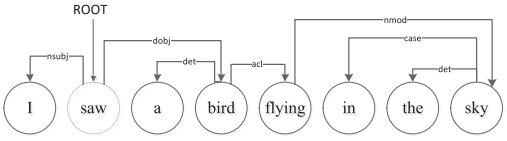


FIGURE 4. Toy example of syntactic dependency. In the figure, each node is a word, and each edge is associated with the type of relations.

FIGURE 4 shows an example syntactic dependency for the sentence "I saw a bird flying in the sky". The syntactic dependency can be represented by 2 types of logic rules. They are "$\pi \rightarrow saw$ represented by Rule 1 with type 'root', and $saw \xrightarrow{dobj} bird$ can be represented by Rule 3 with type 'dobj'. Usually, there is no weight associated with the edges in the syntactic dependency, and p_{ij} is set to 1 for all the rules.

B. EMBEDDING LOGIC RULES INTO RNN

Following, we will show how to modify the input of tradition RNN to embed the logic rules into the traditional model. The key is using the logic rules to eliminate the redundancy connections among the inputs. Here we focus on Rule 1-4 because Rule 5 ($B \vee C \rightarrow A$) can be converted into Rule 4 according to Morgan's Law. Given a current input word A of the input sentence and its matching logic rule $< r_k, \varepsilon_k, p_k >$, the input of RNN is modified according to the type of the rule as follows.

- If r_k belongs to Rule 1, the input of the Logic-RNN is ε_k, π, and A.
- If r_k belongs to Rule 2, the input of the Logic-RNN is ε_k, $\neg$, and A.
- If r_k belongs to Rule 3, the input of the Logic-RNN is ε_k, B, and A.
- If r_k belongs to Rule 4, the input of the Logic-RNN is ε_k, B, C, and A.
- If r_k belongs to Rule 5, the rule is converted to Rule 4 by Morgan's Law, and Rule 4 is used.

The correspondence relationship between each logic rule and the modification of the input structure is shown in FIGURE 5. In the figure, 'A' is the current word of the input sentence, B and C are previous words which are matched and retrieved by the corresponding rule, and π is a placeholder used to align the input of each step. In this paper, π is set to 0 and ε_k is presented using one-hot representation. Note that the weight of the rule is not considered in above embedding approach, which will be considered in mask matrix in the concrete algorithms.

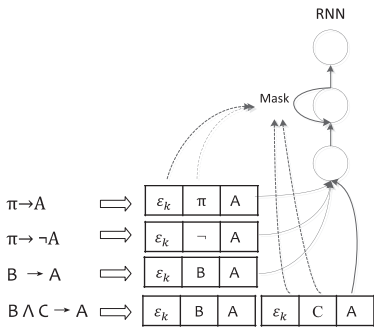


FIGURE 5. The correspondence between the rule and the input of the logic-RNN. The current input is 'A'. 'Mask' is used to filter redundant information of the next time step.

1) LOGIC-RNN

Taking into account the feedback loop problem, we utilized a mask matrix to shield the redundant part of the information, as shown in FIGURE 6. We modified the update of an RNN as follows:

$$h^{(t)} = f(Ux^{(t)} + Wh_c^{(t-1)} + b), \quad (1)$$

$$h_c^{(t)} = f(U'CEM(x^{(t)}, Mask) + W'h_c^{(t-1)} + b'). \quad (2)$$

where f is usually a nonlinear activation function. $U(U')$, $W(W') \in R^{H\times d}$, and $b(b') \in R^H$ are trainable parameters. $h^{(t)} \in R^H$ is the output of the hidden layer, and $h_c^{(t)} \in R^H$ is the hidden layer vector passed to the next time step. To embed the logic rules for the instance, $x_j \rightarrow x_i$ marked as ε_k, a combination of $(\varepsilon_k, x_i, x_j)$ was obtained, and the input was $x^{(t)} = \varepsilon_k^{(t)} \oplus x_j^{(t)} \oplus x_i^{(t)}$, where $\oplus$ indicates the concatenation operation, and π is a virtual variable for simplicity.

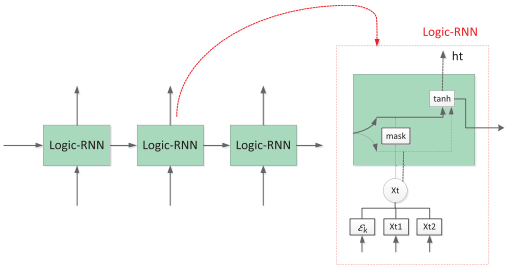


FIGURE 6. Logic-RNN: A modified RNN with embedded logic rules. The inputs masked by the matrix are the corresponding logic rules for every time step. There are 2 data flows: the output of the current time step denoted by the dashed line, and the information to the next time step denoted by the solid line.

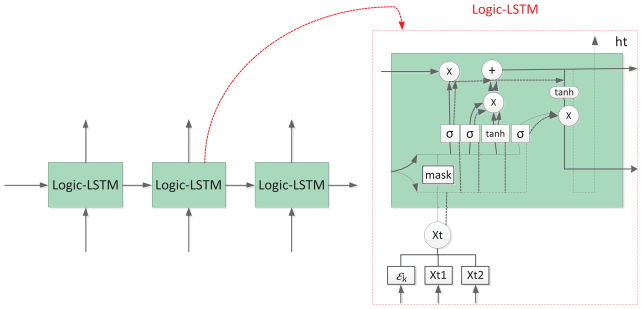


FIGURE 7. Logic-LSTM: A modified LSTM with embedded logic rules. The inputs masked by the matrix are the corresponding logic rules for every time step. There are 2 data flows: the output of the current time step denoted by the dashed line, and the information to the next time step denoted by the solid line.

The 'mask' is a shielding matrix, which is composed of 0 and the relative weight of the rule. The size of the mask matrix is the same as that of one logic rule, i.e. $d * 3$. Here the cell with 0 is used to efficiently remove the redundant information, and non-zero cells (is the weight of rule) reflects the relative importance of the rules. CEM(M_1, M_2) is a function that denotes that the 2 matrices have the same dimensions and are multiplied by the corresponding items. Here, H and d are the dimensionality of the hidden layer and input.

2) LOGIC-LSTM

In order to eliminate the redundant information, a mask matrix is added into Logic-LSTM, similar to that of Logic-RNN.

There are 2 data streams. The first is with a mask matrix that eliminated the redundant information and transferred the useful information to the next time step. The second performs the output along with all the input information and the hidden vector. Here, we modified the update of the LSTM as follows.

$$i^t = \sigma(W_i x^{(t)} + U_i h_c^{(t-1)} + b_i) \quad (3)$$
$$f^t = \sigma(W_f x^{(t)} + U_f h_c^{(t-1)} + b_f) \quad (4)$$
$$o^t = \sigma(W_o x^{(t)} + U_o h_c^{(t-1)} + b_o) \quad (5)$$
$$\tilde{c}^{(t)} = \tanh(W_c x^{(t)} + U_c h_c^{(t-1)} + b_c) \quad (6)$$
$$c^{(t)} = f^{(t)} \odot c_c^{(t-1)} + i^{(t)} \odot \tilde{c}^{(t)} \quad (7)$$
$$h^{(t)} = o^{(t)} \odot \tanh(c^{(t)}) \quad (8)$$
$$i_c^{(t)} = \sigma(W_i' CEM(x^{(t)}, Mask) + U_i' h_c^{(t-1)} + b_i') \quad (9)$$
$$f_c^{(t)} = \sigma(W_f' CEM(x^{(t)}, Mask) + U_f' h_c^{(t-1)} + b_f') \quad (10)$$
$$o_c^{(t)} = \sigma(W_o' CEM(x^{(t)}, Mask) + U_o' h_c^{(t-1)} + b_o') \quad (11)$$
$$\tilde{c_c}^{(t)} = \sigma(W_c' CEM(x^{(t)}, Mask) + U_c' h_c^{(t-1)} + b_c') \quad (12)$$
$$c_c^{(t)} = i_c^{(t)} \odot c_c^{(t-1)} + i_c^{(t)} \odot \tilde{c_c}^{(t)} \quad (13)$$
$$h_c^{(t)} = o_c^{(t)} \odot \tanh(c_c^{(t)}) \quad (14)$$

Here, σ is the logistic sigmoid function. The operator $\odot$ is the pointwise multiplication of the 2 vectors. $h^{(t)} \in R^H$ is the output of the hidden layer, and $h_c^{(t)} \in R^H$ is the hidden layer vector passed to the next time step. $i^t(i_c^{(t)})$, $f^t(f_c^{(t)})$, $o^t(o_c^{(t)})$ and $\tilde{c}^t(\tilde{c}_c^{(t)})$ are the input gate, the forget gate, the output gate and the memory cell activation vector at a given time step, respectively. These are all the same size as the hidden vector $h^{(t)} \in R^H$. $W_i(W_i')$, $W_f(W_f')$, $W_o(W_o')$, $W_c(W_c') \in R^{H \times d}$, $U_i(U_i')$, $U_f(U_f')$, $U_o(U_o')$, $U_c(U_c') \in R^{H \times H}$ are trainable parameters. To embed the logic rules for instance $x_j \rightarrow x_i$ marked as ε_k, we then obtained the combination (ε_k, x_i, x_j, π) and the input $x^{(t)} = \varepsilon_k^{(t)} \oplus x_j^{(t)} \oplus x_i^{(t)} \oplus \pi$, where $\oplus$ indicates the concatenation operation, and π is a virtual variable for simplicity. Mask is a shielding matrix that is composed of 0 and 1, where the shape of $1 * d$ is used to efficiently remove the redundant information. CEM(M_1, M_2) is a function where 2 matrices with the same dimension are multiplied by the corresponding items. Here, H and d are the dimensionality of the hidden layer and the input.

IV. EXPERIMENTS

To verify the effectiveness and the universality of the proposed method, we conduct experiments on two different tasks, sentiment classification and named entity recognition and an additional experiment on sentiment classification with a small number of training samples. To obtain a fair comparison, only syntactic dependency extracted from the input sentences are used as prior knowledge, and no external knowledge base is used in the experiments. Here, we use the Stanford CoreNLP tool to obtain syntactic dependencies.

In all the following experiments, RNN, LSTM, and BI-LSTM are used as the baseline method, because of these three methods are still among the state-of-the-art algorithms for the sentiment classification and NER task. Though there are a lot of algorithms are proposed based on these models, the structures remain the same, and our embedding strategy for RNN, LSTM, and BI-LSTM can be easily extended to other models.

A. SENTIMENT CLASSIFICATION

1) EXPERIMENT SETTING

For the sentiment analysis task, we process $x^{(t)} = \varepsilon_k^{(t)} \oplus x_j^{(t)} \oplus x_i^{(t)}$ as the input of the network structure, and then make the output of the last hidden layer of logic-RNN as semantic feature, label as h_n. Finally, we use the semantically encoded expression vector h_n as the input of the softmax regression

function to obtain a large probability distribution for each sentiment category. The probability of the category of c is:

$$P_j(y = c \mid h_n) = soft\max(V_c^T h_n + b_c) = \frac{\exp(V_c^T h_n + b_c)}{\sum_{i=1}^{C} \exp(V_i^T h_n + b_c)} \quad (15)$$

Here, $P_j(y = c \mid h_n)$ is the category of the j-th sample, V, b_c are softmax regression parameters, and C is the number of categories.

We use cross entropy as the objective function of model training, add an L_2 regularization term to prevent over-fitting of the model, and θ is all parameters of the model.

$$J(\theta) = -\frac{1}{M}\sum_{j=1}^{M}\sum_{i=1}^{C} \log P_{ji} + \frac{\lambda}{2}||\theta||^2 \quad (16)$$

Here, j is the j-th sample, M is the number of samples, i is the i-th category.

The methodology was tested on the document-level sentiment classification task, using the real-world datasets Yelp 2013 and Dangdang (a famous e-commerce platform in China). We pre-processed and split the datasets as described by Tang *et al.* [3]. The statistic information of the two data sets are shown in TABLE 1 and TABLE 2.

TABLE 1. The details of yelp2013 dataset.

Dataset	1	2	3	4	5	Total
Training Set	2179	5130	11989	26057	17166	62521
Verification Set	246	669	1491	3207	2159	7772
Testing Set	299	751	1607	3567	2447	8671

TABLE 2. The details of Dangdang dataset.

Dataset	1	2	3	4	5	Total
Training Set	420	966	614	596	1212	3808
Verification Set	154	324	194	208	388	1268
Testing Set	152	330	192	196	400	1270

The Acc (Accuracy) [28] and the MSE (mean squared error) [29] were used as the evaluation metrics. Note that the Acc is the main metric for the performance of the algorithm, and MSE is used to measure the convergence of the algorithm. The parameters are tuned on the validation set according to the classification accuracy. The learning rate was set to 0.01, the batch size was set to 64, the number of hidden units was set to 120, and the parameter λ of L_2 regularization was set to 1e-4. For model initialization, we learned the 50-dimensional word embeddings with Skip-Gram [30] on the dataset itself. Additionally, we initialized the rule tag ε_k and the other parameters, except for those parameters for the recurrent neural networks from a uniform distribution $U(-0.1, 0.1)$. The pre-trained word embeddings were allowed to be modified during training. Finally, the Adagrad [31] optimizer was used to train the model.

2) RESULTS

Table 3 shows the comparison between the proposed approach and the baselines. The classification accuracy and MSE of our models were compared with the baseline models.

TABLE 3. Sentiment classification on Yelp 2013 dataset and Dangdang dataset.

Model	Acc of yelp2013	MSE of yelp2013	Acc of Dang-dang	MSE of Dang-dang
RNN	52.9	0.669	27.4	4.548
Logic-RNN	**54.7**	**0.646**	**31.2**	**3.757**
LSTM	54.3	0.651	32.1	3.641
Logic-LSTM	**56.2**	**0.656**	**33.7**	**3.262**
BI-LSTM	55.3	0.664	32.9	3.416
Logic-BI-LSTM	**56.3**	**0.649**	**35.6**	**3.409**

The results were averaged over 10 runs of the algorithms.

Comparing the RNN with the Logic-RNN, it can be seen from TABLE 3 that the accuracy of Logic-RNN is nearly 2% higher than that of RNN on yelp2013 dataset, and the MSE of Logic-RNN is lower than that of RNN. The accuracy of Logic-RNN is nearly 4% higher than that of RNN on Dangdang dataset, and the MSE of Logic-RNN is lower than that of RNN. The effect of adding the logic rules into RNN is better than original RNN.

Comparing the LSTM with the Logic-LSTM, it can be seen from TABLE 3 that accuracy of Logic-LSTM is nearly 2% higher than that of LSTM on yelp2013 dataset. The accuracy of Logic-LSTM is 1.6% higher than that of LSTM on Dangdang dataset, and the MSE of Logic-LSTM is lower than that of LSTM. This indicates that the logic rules are able to improve the representation of LSTM.

Comparing the BI-LSTM with the Logic-BI-LSTM, it can be seen from TABLE 3 that the accuracy of Logic-BI-LSTM is nearly 1% higher than that of BI-LSTM on the yelp2013 dataset, and the MSE of Logic-BI-LSTM is lower than that of BI-LSTM. The accuracy of Logic-BI-LSTM is 2.7% higher than that of BI-LSTM on Dangdang dataset, and the MSE of Logic-BI-LSTM is lower than that of BI-LSTM. This result not only shows that the BI-LSTM is able to learn the context information of text but also shows the Logic-BI-LSTM can learn more and improve the emotional representation of the BI-LSTM.

The comparison above shows that the embedded rule always improves the performance of the baseline methods in

both Acc and MSE. These results indicate the effectiveness of the embedding strategy on the sentiment classification task.

B. NAMED ENTITY RECOGNITION

1) CONCRETE ALGORITHM

We treat named entity recognition as a sequence labeling process for an input sequence, $X = (x_1, x_2, \cdots, x_T)$. Here, x_t is the t-th word of the sequence, T is the length of the sequence. The task of the name entity recognition is prediction the label $y = (y_1, y_2, \cdots, y_T)$ of each input x_t, and a corresponding output sequence is $y = (y_1, y_2, \cdots, y_T)$. The objective function of named entity recognition is follow, and θ is the parameter of the model. Here, $y_t = softmax(h_n)$ and h_t is the hidden layer of Logic-RNN as semantic feature.

$$s(y_1, y_2, \cdots, y_T \mid X; \theta) = \prod_{t=1}^{T} s(y_t \mid x; \theta) \tag{17}$$

2) SETUP

In this experiment, the syntactic dependency is used as the prior knowledge and no other additional features (such as pos tag, char, and pre-trained word vectors) are used in the model. We set learning rate as 0.0125, batch size as 256, number of hidden units as 290, input keep probability as 0.58, and output keep probability as 0.88. For model initialization, we initialized the lookup table and the rule tag ε from the uniform distribution U(−1.0,1.0) and the other parameters. Note that the parameters of the recurrent neural networks were not included for a uniform distribution U(−0.1,0.1).

We evaluated our proposed method on the CoNLL-2003 dataset for named entity recognition. The dataset contains 14,987 / 3,466 / 3,684 sentences and 204,567 / 51,578 / 46,666 tokens in the train/dev/test sets, respectively. The dataset includes 4 entity types: Person (PER), Location (LOC), Organization (ORG), Miscellaneous (MISC), and the non-entity elements tagged as (O). We used the F1-score to evaluate the performance of the models.

3) RESULTS

TABLE 4 shows the performance on the NER task. Comparing the RNN with the Logic-RNN, it can be seen from TABLE 4 that the F1 of Logic-RNN was 78.8, which higher than the RNN on the CoNLL-2003 dataset. The F1 of Logic-LSTM is 79.3, which is higher than that of LSTM. The F1 of Logic-BI-LSTM is 81.6, which is higher than that of BI-LSTM.

These results show that the embedding strategy always improves the performance of the baseline algorithms in terms of Acc. Though the MSE of RNN is smaller than that of RNN, these results indicate the effectiveness of the embedding strategy on the name entity recognition task.

TABLE 4. NER results on the CoNLL-2003 dataset.

Model	F1
RNN	76.9
Logic-RNN	**78.8**
LSTM	77.6
Logic-LSTM	**79.3**
BI-LSTM	80.2
Logic-BI-LSTM	**81.6**

The results were averaged over 10 runs of the algorithms.

C. SMALL SAMPLE EXPERIMENT

To further verify the performance of the proposed approach on the small sample data set, we tested the algorithms on two randomly sampled sub-datasets from yelp2013. In detail, 5%, 10% samples were used as the training set in the experiments, the other settings of the experiments, including the verification and testing set are the same as the default setting of the experiments. The results are shown in TABLE 5 and TABLE 6.

TABLE 5. Sentiment classification results on the Yelp 2013 dataset, with 10% samples were used.

Model	Acc	MSE
RNN	45.0	**0.809**
Logic-RNN	**48.5**	0.841
LSTM	47.5	0.807
Logic-LSTM	**49.6**	**0.793**
BI-LSTM	47.0	0.808
Logic-BI-LSTM	**50.1**	**0.775**

TABLE 6. Sentiment classification results on the Yelp 2013 dataset, with 5% of the samples were used.

Model	Acc	MSE
RNN	42.3	1.072
Logic-RNN	**44.6**	**0.879**
LSTM	43.8	0.931
Logic-LSTM	**47.5**	**0.839**
BI-LSTM	46.1	**0.843**
Logic-BI-LSTM	**48.2**	0.844

TABLE 5 shows that the accuracy of Logic-RNN is 3.5% higher than that of RNN, though the MSE of RNN is smaller than that of the Logic-RNN. Similarly, the accuracy of Logic-LSTM is 2.1% higher than that of LSTM, the accuracy of Logic-BI-LSTM is 3.1% higher than that of BI-LSTM.

TABLE 6 shows that the accuracy of Logic-RNN is 2.3% higher than that of RNN, the accuracy of Logic-LSTM is 3.7% higher than that of LSTM, the accuracy of Logic-BI-LSTM is 2.1% higher than that of BI-LSTM. In summary, the logic-RNN outperforms RNN.

FIGURE 8 shows an important observation that the logic-RNN achieves the highest improvement when only 5% samples are used as the training set. This reflects that the logic rules provide additional information to the model training, and such information is important for the sample lacking

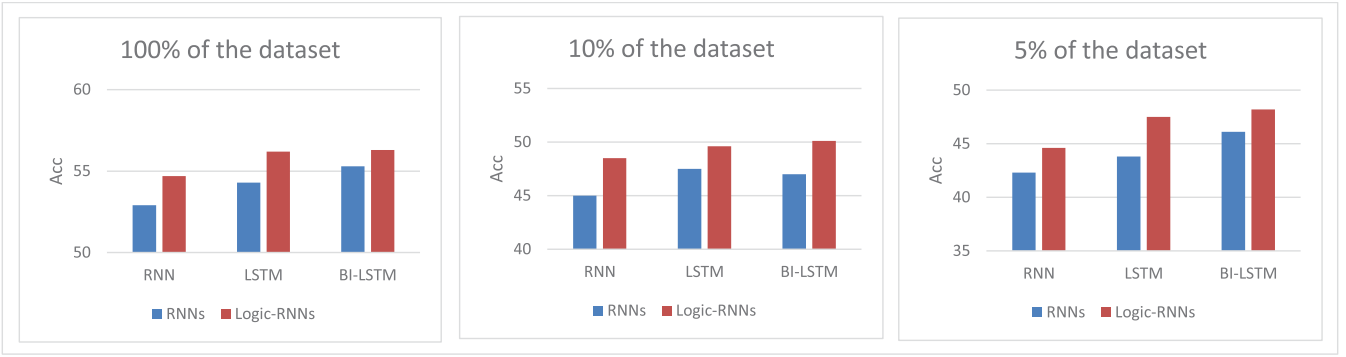


FIGURE 8. Sentiment classification on the Yelp 2013 dataset, with varying number of samples.

cases. This property not only verifies the motivation of our work but also shows the fitness of the model on the real-world applications with a small number of training data.

V. CONCLUSION AND FUTURE WORK

We proposed a method to embed logic rules into an RNN by using a mask metric to remove redundant feedback. Results showed that the logic rules were able to present prior knowledge from different domains, and the proposed embedding approach worked well on a variety of tasks.

Future work should include applying logic-RNN to other tasks, such as Q&A, machine translations, and speech recognition, as well as embedding the rules into other recurrent neural networks, like GRU [32], and extending our approach from the first-order logic rules to the higher-order logic rules.

REFERENCES

[1] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2013, pp. 6645–6649.

[2] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio. (2014). "On the properties of neural machine translation: Encoder-decoder approaches." [Online]. Available: https://arxiv.org/abs/1409.1259

[3] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proc. EMNLP*, 2015, pp. 1422–1432.

[4] J. P. C. Chiu and E. Nichols. (2015). "Named entity recognition with bidirectional LSTM-CNNs." [Online]. Available: https://arxiv.org/abs/1511.08308

[5] Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. Xing. (2016). "Harnessing deep neural networks with logic rules." [Online]. Available: https://arxiv.org/abs/1603.06318

[6] X. Zhu, H. Guo, S. Mohammad, and S. Kiritchenko, "An empirical study on the effect of negation words on sentiment," in *Proc. ACL*, vol. 1, 2014, pp. 304–313.

[7] P. Mirowski and A. Vlachos. (2015). "Dependency recurrent neural language models for sentence completion." [Online]. Available: https://arxiv.org/abs/1507.01193

[8] B. M. Lake, R. Salakhutdinov, and J. B. Tenebaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[9] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: A collaboratively created graph database for structuring human knowledge," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2008, pp. 1247–1250.

[10] G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[11] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow. (2011). "The anatomy of the Facebook social graph." [Online]. Available: https://arxiv.org/abs/1111.4503

[12] T. Mikolov, S. Kombrink, A. Deoras, L. Burget, and J. Cernocky, "RNNLM-recurrent neural network language modeling toolkit," in *Proc. ASRU Workshop*, 2011, pp. 196–201.

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

[14] K. S. Tai, R. Socher, and C. D. Manning. (2015). "Improved semantic representations from tree-structured long short-term memory networks." [Online]. Available: https://arxiv.org/abs/1503.00075

[15] X. Wang, Y. Liu, C. Sun, B. Wang, and X. Wang, "Predicting polarities of tweets by composing word embeddings with long short-term memory," in *Proc. ACL*, vol. 1, 2015, pp. 1343–1353.

[16] J. Xu, D. Chen, X. Qiu, and X. Huang. (2016). "Cached long short-term memory neural networks for document-level sentiment classification." [Online]. Available: https://arxiv.org/abs/1610.04989

[17] J. Hammerton, "Named entity recognition with long short-term memory," in *Proc. 7th Conf. Natural Lang. Learn. (HLT-NAACL)*, vol. 4, 2003, pp. 172–175.

[18] Z. Huang, W. Xu, and K. Yu. (2015). "Bidirectional LSTM-CRF models for sequence tagging." [Online]. Available: https://arxiv.org/abs/1508.01991

[19] X. Ma and E. Hovy. (2016). "End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF." [Online]. Available: https://arxiv.org/abs/1603.01354

[20] 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.

[21] G. G. Towell and J. W. Shavlik, "Knowledge-based artificial neural networks," *Artif. Intell.*, vol. 70, nos. 1–2, pp. 119–165, 1994.

[22] A. S. A. Garcez and G. Zaverucha, "The connectionist inductive learning and logic programming system," *Appl. Intell.*, vol. 11, no. 1, pp. 59–77, Jul. 1999.

[23] T. Rocktäschel, M. Bošnjak, S. Singh, and S. Riedel, "Low-dimensional embeddings of logic," in *Proc. ACL Workshop Semantic Parsing*, 2014, pp. 45–49.

[24] T. Rocktäschel, S. Singh, and S. Riedel, "Injecting logical background knowledge into embeddings for relation extraction," in *Proc. HLT-NAACL*, 2015, pp. 1119–1129.

[25] D. Koller, N. Friedman, and F. Bach, *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA, USA: MIT Press, 2009.

[26] G. F. Cooper, "NESTOR: A computer-based medical diagnostic aid that integrates causal and probabilistic knowledge," Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, 1984.

[27] H. E. Brady, "Causation and explanation in social science," in *The Oxford Handbook of Political Science*. Oxford, U.K.: Oxford Univ. Press, 2008.

[28] D. Jurafsky, *Speech & Language Processing*. London, U.K.: Pearson Education India, 2000.

[29] Q. Diao, M. Qiu, C.-Y. Wu, A. J. Smola, J. Jiang, and C. Wang, "Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS)," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 193–202.

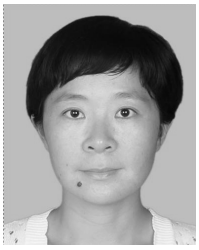
[30] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.
[31] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Feb. 2011.
[32] K. Cho *et al.* (2014). "Learning phrase representations using RNN encoder-decoder for statistical machine translation." [Online]. Available: https://arxiv.org/abs/1406.1078

BINGFENG CHEN received the B.S. and M.S. degrees in computer science from the Guangdong University of Technology, in 2007 and 2010, respectively, where he is currently an Assistant Researcher with the School of Computer. His research interests include machine learning, data mining, and natural language processing.

ZHIFENG HAO received the B.S. degree in mathematics from Sun Yat-sen University, in 1990, and the Ph.D. degree in mathematics from Nanjing University, in 1995. He is currently a Professor with the School of Computer, Guangdong University of Technology, and with the School of Mathematics and Big Data, Foshan University. His research interests include various aspects of algebra, machine learning, data mining, and evolutionary algorithms.

XIAOFENG CAI received the B.S. and M.S. degrees from the Guangdong University of Technology, in 2015 and 2018, respectively. Her current research interests include machine learning and natural language processing.

RUICHU CAI received the B.S. degree in applied mathematics and the Ph.D. degree in computer science from the South China University of Technology, in 2005 and 2010, respectively. He was a Visiting Student with the National University of Singapore, from 2007 to 2009, and a Research Fellow with the Advanced Digital Sciences Center, Illinois at Singapore Pte. Ltd., from 2013 to 2014. He is currently a Professor with the School of Computer, Guangdong University of Technology. His research interests include a variety of different topics, including causality, machine learning, and their applications.



WEN WEN received the B.S. degree in applied mathematics and the Ph.D. degree in computer science from the South China University of Technology, in 2003 and 2008, respectively. She is currently an Associate Professor with the School of Computer, Guangdong University of Technology. Her research interests include a variety of different topics, including machine learning, graph embedding, and sequential data analysis.

JIAN ZHU received the B.S. degree in mathematics from Sun Yat-sen University, China, in 2005, and the Ph.D. degree in computer science from the University of Macau, in 2013. He is currently an Assistant Professor with the Guangdong University of Technology. His research interests include computer graphics, computer vision, and machine learning techniques.

GUANGQIANG XIE received the B.S. degrees from Northeast Electric Power University, in 2001, and the Ph.D. degree from the Guangdong University of Technology, in 2013, where he is currently an Associate Professor. His research interests include machine learning and cloud computing.