IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 4, 2019, accepted April 8, 2019, date of publication April 18, 2019, date of current version April 29, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2911889

Active Learning for Uneven Noisy Labeled Data in Mention-Level Relation Extraction

WEI YULIANG, XIN GUODONG, WANG WEI, AND WANG BAILING
Harbin Institute of Technology, Harbin 264209, China

Corresponding author: Wang Bailing (wbl@hit.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB0800802, and in part by the Shandong Key Research and Development Plan under Grant 2016ZDJS01A04 and Grant 2017CXGC0706.

ABSTRACT Mention-level relation extraction (mRE) plays an important role in extracting relational information from short texts such as those exchanged in a social network. Deep learning (DL) has made remarkable achievements; the main problem encountered with DL in mRE is a lack of training samples. In this paper, we present a design for a quick sample-marking method. First, we construct an uneven noisy labeled data (UNLD) set using a pattern matching algorithm, and then a relabeling framework is put forward for modifying the UNLD. With regard to the accuracy, the recall rates of categories with sufficient samples increased from 0.4 to nearly 1 using the relabeling framework. We have released our code and other resources for further research (https://github.com/curtainsky/UNLD).

INDEX TERMS Relation extraction, active learning, text mining, deep learning.

I. INTRODUCTION

Web Ontology Language (OWL) is a method of storage for knowledge graphs, and triples are the basic storage structures in OWL. Extracting relational triples from social network data is an important step in establishing a social network knowledge graph. The main task of mention-level relation extraction (mRE) is to identify the essential relations in a sentence from the given context [1]. Social network text usually does not exceed 140 characters, and in this context, mRE is the extraction of triplets from social networks. A relation can be expressed by a triple: ($entity_1$, $relation$, $entity_2$), or (E_1, R, E_2) for short. The complexity of mRE is reflected in three ways: (1) Different entities in a sentence can have different relations; (2) relations between two entities may be different in different sentences; (3) the definitions of a specific mRE may be different in different scenarios. For example, the sentence ''Joy was born in New York, a famous city in the United States'' contains two relations, (Joy, BornIn, United States) and (New York, CityOf, United States). In the sentence ''Joy is the president of the United States,'' however, the relation between Joy and United States is LeaderOf, whereas in another sentence, ''Joy loves Snoop Dogg, a rapper from the United States,'' Joy has no relation with United States. Owing to the complexity of relation extraction (RE), there is as yet no automatic method for achieving acceptable accuracy.

mRE is formulated as a multi-classification problem, and each category corresponds to a specific relation type. Previous methods for mRE can be broadly classified into two types: feature-based methods and kernel-based methods. Feature-based methods depend on feature engineering [2]; thus, they are dependent on specific issues and are difficult to apply to other issues. In kernel-based methods, kernel functions are designed to compute similarities between two relation entities, and then support vector machines (SVMs) are employed for classification [3]. Kernel-based methods effectively turn feature engineering into a similarity computation for the purpose of increasing the applicability of algorithms. In the past two years, deep learning (DL) algorithms have been widely used in RE [4], [5]. In contrast to the kernel-based methods, DL algorithms automatically learn patterns from labeled samples without a manually designed grammar kernel. With the passage of time, the advantages of deep learning algorithms have gradually become clear.

The main problem with DL methods is that the training of a neural network requires a large number of samples but the most current training data sets do not meet this need. To avoid the leaking of training samples, current DL methods for mRE mainly use pre-trained word embedding with word2vec [6]. The fact that the optimization goal of generating word embeddings is not consistent with that of mRE

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

can cause unexpected problems. The main contribution of this paper is therefore to describe a method for constructing training samples quickly and effectively. The method first uses previous algorithms to quickly construct an uneven noisy labeled data (UNLD) set, and then a labeling framework is proposed that automatically optimizes the UNLD for reducing the noise of the samples. In the experimental part, we apply the framework of active learning to relation extraction. When the iteration threshold is set to 85%, the sampling accuracy is as high as 91%.

The paper is organized as follows. Section II introduces the details of the UNLD set and proposes a labeling framework. Section III analyzes the conditions of the labeling framework, and a deep learning model based on an attention model is proposed for mRE. In Section IV, experimental results demonstrate that our framework optimizes the UNLD effectively. Section V summarizes the findings and describes future work to be done.

II. RELATED WORKS

The deep learning model improves the accuracy of the algorithm in many fields of natural language processing. While in mRE, because of the large number of relation types required in building knowledge graph, the public sample set cannot satisfy the specific relation extraction. The proposed algorithm is difficult to directly apply the deep learning algorithm to production, and can only prove the advantages of deep learning on a small data set. At present, the core problem in expanding the scope of use of relation extraction algorithm is how to quickly obtain labeled samples. The mainly research methods for constructing relation extraction samples are semi-supervised learning and Distant Supervision.

Semi-supervised learning for sample tagging relies mainly on traditional learning methods. Sun et al. uses clustering method: first marks some data, then clusters the large sample data according to the similarity of sentences, and then marks the pairs of entities in the category [7]. This method can reduce the number of samples for labeling, while the sentence similarity calculation is rough, and the effect is not good when the relationship type is increased. Chen et al. investigate a graph based semi-supervised learning algorithm for relation extraction [8]. It represents labeled and unlabeled examples and their distances as the nodes and the weights of edges of a graph, and tries to obtain a labeling function to satisfy two constraints: 1) it should be fixed on the labeled nodes, 2) it should be smooth on the whole graph. Erkan et al. is based on the analysis of the paths between two proteins names in the depending parse trees of the sentences [9]. They define two separate similarity functions (kernels) based on cosine similarity and edit distance among the paths between the protein names. The main problem of the traditional semi-supervised algorithm is that in the semi-supervised iterative process, the labeled samples are not detected. When the accuracy of the sample markers is not high, the iterative results are unstable and susceptible to the error sample marking.

Distant Supervision (DS) is under the assumption that if two entities have a relationship in a KB, then all sentences mentioning those entities express the same relation [10]. This approach works well in generating large amounts of training instances, the DS assumption does not hold in all cases [11], [12]. The input to the DS algorithm is the pair of entities that need to be tagged and all the sentences in the corpus that contain the two entities. Because the corpus is easy to obtain, there are a large number of papers around DS to construct a relational extraction set in recent years [13]. Although DS algorithm can only label a large number of corpus, while the realtion has no meaning for a single sentence, the labeling samples acquired by DS cannot be applied to mRE.

When constructing domain knowledge graph from texts, it is necessary to customize a large number of relations according to domain problems. Because of the certain ambiguity between relations, such as ''work'', ''leadership'', ''responsibility'' can simultaneously represent the relation between people and institutions, in the manual marking process, there is high missing labels and mislabeling than other. Samples that are automatically labeling using template matching methods also contain a high error rates. Therefore traditional algorithms are not suitable for sample tags of mRE. In this paper, we find that deep learning has self-correction ability for mislabeled samples in classification problems, and designs a marker framework for active learning, which greatly improves the accuracy of sample markers.

III. UNEVEN NOISY LABELED DATA

There are many outstanding ways of accomplishing mRE, among which pattern matching methods are the most straightforward technique for obtaining samples. The first pattern matching algorithm was proposed by [14], named dual iterative pattern relation expansion (DIPRE). For a sentence $[w_1, \ldots, w_{i-1}, E1, w_{i+1}, \ldots, w_{j-1}, E2, w_{j+1}, \ldots w_n]$, the DIPRE splits the sentence into [⟨prefix⟩, $E1$, ⟨middle⟩, $E2$, ⟨suffix⟩], in which ⟨prefix⟩ is equal to $[w_1, \ldots, w_{i-1}]$, and ⟨middle⟩ is $[w_{i+1}, \ldots, w_{j-1}]$, and ⟨suffix⟩] is $[w_{j+1}, \ldots w_n]$. DIPRE first finds the pattern (prefix, middle, suffix) through $E1$ and $E2$, and then uses the pattern to find new entities. An improved system was proposed by [15], named SnowBall. In SnowBall, entities are represented as tags, such as *PER*, *ORG*, and *LOC*. The core of pattern matching algorithms is the length of affixes. If an affix is too long, the recall rate will be low and the accuracy high, and vice versa if the affix is too short. This method is very useful for constructing small sample sets having high accuracy.

For convenience, we call sample sentences without relations negative samples (NAs); conversely, samples with relations are called positive samples. Before building the UNLD set, we first introduce the method for constructing positive samples and NAs. For a new mRE task, SnowBall is adapted for collecting positive samples with high accuracy. As the affixes need to be long enough to guarantee accuracy, it is difficult to obtain a sufficiently large number of labeled samples

using SnowBall. However, training word embedding requires a large number of samples, and after the positive samples are obtained, the training set still requires a sufficient number of NAs. Put simply, an NA sample can be quickly labeled in two ways: (1) Sentences without entities can be labeled as NAs; (2) sentences containing no relation entity pairs can be labeled as NA samples. Then the training set will have a limited number of positive samples and a large number of NAs in addition to many unlabeled samples.

Traditional semi-supervised algorithms, such as co-training [16], first construct multiple classifiers by training the labeled samples and then mark the unlabeled samples with the predicted results of these classifiers. This approach has two problems: (1) As the training samples are extremely uneven, a large number of samples are classified as NA; (2) for deep learning, the more samples that participate in the training, the better the network optimization. Therefore, we use the unlabeled samples and the labeled samples in training at the same time. In order to facilitate the training, the unlabeled samples are marked as NA. The training set is the UNLD set, which combines labeled samples with unlabeled samples, as shown in Fig. 1. The noise in the UNLD is reflected in the incorrect labels of the unlabeled samples and the error rate of SnowBall.

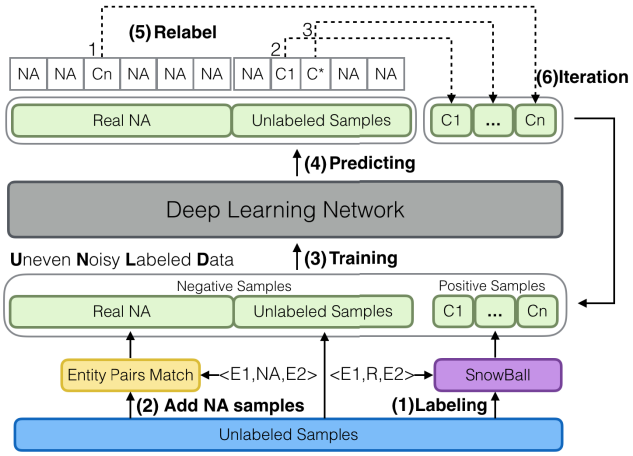

FIGURE 1. The framework for relabeling UNLD.

The framework for relabeling the UNLD is shown in Fig. 1: (1) Labeling. Users need to first obtain a small number of positive labeled samples by manual or pattern matching, likes SnowBall; (2) Add NA samples. Select a certain number of unlabeled samples to label as negative samples (NA); (3) Training. The positive samples, gathered by step 1, combining with the negative samples, gathered by step 2, are as training set UNLD. UNLD means the training set is uneven and noisy. (4) Predicting. The deep learning (DL) network for relation extraction is training by UNLD. (5) Relabel. For all of the samples in NA which are predicting as positive samples by DL are called reNA. The reNA is the samples set needed to re-label, and the framework re-labels the reNA samples as the corresponding categories predicted by DL. (6) Iteration. The framework repeats 3 to 5 steps until convergence. For example, if the label of a particular sample is NA, but the prediction of the DL algorithm is C1, then the framework re-marks this sample into the C1 category, as shown in Fig. 1, line 2. According to the iterative process, line 1 in Fig. 1 is a significant error in relabeling, which needs to be avoided during the iteration. In the next section, we discuss the conditions for this relabeling framework.

IV. DEEP LEARNING FOR UNLD

In this section, we describe the assumptions for deep learning and provide the conditions of the relabeling framework for the UNLD, and then we introduce some optimization tricks. Finally, we introduce the deeplearning network for mRE.

A. DEEP LEARNING ASSUMPTIONS

Deep learning networks have been shown to be capable of auto-optimizing features based on samples [17]. For example, the kernels of a convolutional neural network (CNN) can be used as extraction matrices of implicit features. The network can generalize unseen feature combinations better with the low-dimensional dense embedding that is learned for sparse features [18]. The most significant advantage of deep learning is that it summarizes features from data automatically rather than relying on manual feature engineering.

With regard to the multi-classification problem, DL can be divided into two components: encoding and classification. When the classification step or the output layer uses a softmax layer, the encoding step is actually trying to map all the samples into a linearly separable space. We use x to represent a training sample, S the sample space, and L a linearly separable space. Then, the encoding step can be represented by a map function, $E : (x) \rightarrow x', x \in S, x' \in L$. As the experiment results in Fig. 2, the ''distance'' between samples belonging to different categories in space L is as large as possible. Although ''distance'' is a vague concept, it can be roughly estimated using the centers of gravity of the categories. The data are generated from four random normal distribution.

Fig. 2(a) shows the vector distribution of the raw sample vectors, and Fig. 2(b) shows the distribution of hidden feature vectors, which are transformed by a multi-layer perceptron (MLP). The background colors show the dividing lines between different categories. In Fig. 2(a), the distances between samples in different categories are small, whereas in Fig. 2(b), the samples are no longer clustered in the center but scattered around the categories to which they belong. The reason is that the goal of optimizing DL is to keep the distinctions between categories as large as possible. The samples near the dividing line are close together both in their raw space and in the implicit feature space, as shown in box 1 and box 2 of Fig. 2. In active learning (AL), the samples near the dividing line and the incorrectly predicted samples need to be examined [19]; these are called hard samples. Hard samples are very important for ascertaining the suitability of the classification model. For instance, if there are essentially

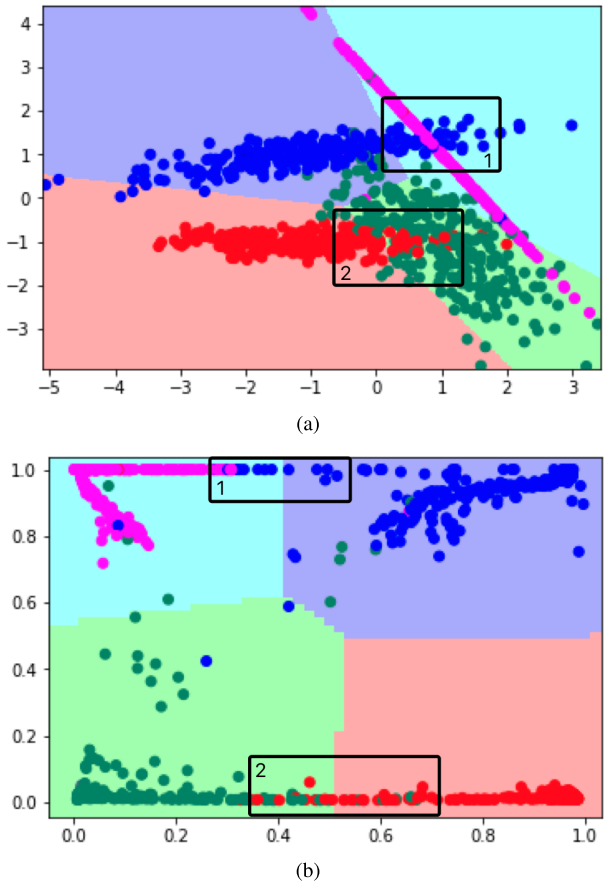

FIGURE 2. Example of hidden features. (a) Raw samples in *S*. (b) Hidden features in *L*.

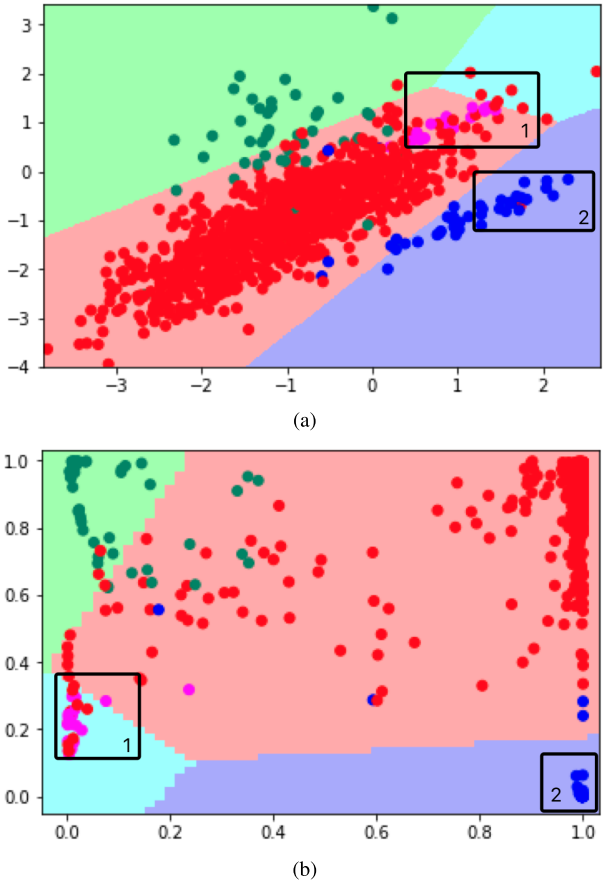

FIGURE 3. Example of UNLD hidden features. (a) Raw samples. (b) Hidden features.

no errors in the labels of the hard samples, this means that the classification algorithm is still inadequate, and the network structure should be further adjusted based on these hard samples. As it is difficult to calculate the dividing line in high-dimensional space, the entropy of the output layer is usually used to estimate the samples that need to be examined.

In the UNLD, the number of NA samples is much greater than those of the other categories. Thus, the encoding model moves easily distinguishable samples to the edge (Fig. 3, box 2), and the indistinguishable samples are gathered near the dividing line (Fig. 3, box 1). Ambiguous samples that have insufficient features are more likely to be classified as NA because of the huge number of NA samples. For the UNLD, we obtained two inferences from the results of many trials: (1) Samples with features that are less distinctive tend to be classified as NA; (2) the accuracy of non-NA categories is rather high.

It is difficult to determine whether a given UNLD set and the encoding network are consistent with these two inferences because the true sample labels are unknown. However, there are two ways to identify whether a non-NA category satisfies inference 2. For convenience, we represent the accuracy of one non-NA category without unlabeled samples as *AWN*. *AWN* can be used to estimate the true accuracy of each non-NA category. If the *AWN* value for one category exceeds a threshold h (0.95, for example), then the confidence in the real labels of unlabeled samples that are predicted into that category is strong. A high *AWN* value means that only samples with strong features are classified into this category. Another way to judge whether the the iterations can be processed is to check the hard samples that need manual intervention. Although manual intervention is required, sampling methods can be used to reduce the number of samples that need to be checked.

B. UNLD TRAINING ITERATION PROCESS

Let X represent the UNLD set, and let Y be the labels of X. E denotes the encoding network, and the *AWN* threshold is h. The process for one iteration is as follows:

(1) Train encoding network $E(X, Y)$ to convergence.

(2) Calculate *AWN* for each category.

(3) For the categories whose *AWN* values are greater than h, modify the labels of the NA samples whose predicted labels are those categories, setting them to the corresponding labels.

When training an uneven data set, it can easily happen that all samples are classified into the category having the greatest number, such as NA. To resolve this problem, we reduce the weight of the NA samples. Moreover, we generate several

initial values and choose the value having the highest accuracy at the beginning. Once the samples are all classified into one category, the iteration process is stopped, and the initialization is re-randomized. Samples with large entropy (hard samples) can be used as a reference for evaluating the encoding network. If the predicted labels of these samples have high accuracy, then our iterative algorithm can improve the recall rate with the accuracy guaranteed.

C. LANGUAGE MODEL AND ENCODING NETWORK

As the network design described in this paper does not include named entity extraction, we use CoreNLP [20] to obtain the entities in the sentence before training. W is the set of words except entities. The entity word set is denoted as E, and each entity has a type label; e.g., *PER*, *ORG*, and *LOC* represent person, organization, and location, respectively. Sentence s is a sequence of words that contains entities; for instance, $s = [w_1, w_2, e_1, w_3, e_2, e_3]$, $w_i \in W, e_i \in E$. The set of all relations is denoted by R. The RE is a mapping function: $F(s, e1, e2) \rightarrow R$, where s is a sentence and $e1, e2 \in E$.

From the examples given at the beginning of this paper, we can see that the relation in a given sentence can differ owing to the differences in entity pairs. Thus, the vector input to the encoding network needs to be demonstrated through the sentence and the entity pair. Now, we provide the structure diagram for the encoding network and describe it in detail. The structure of the DL network is shown in Fig. 4.

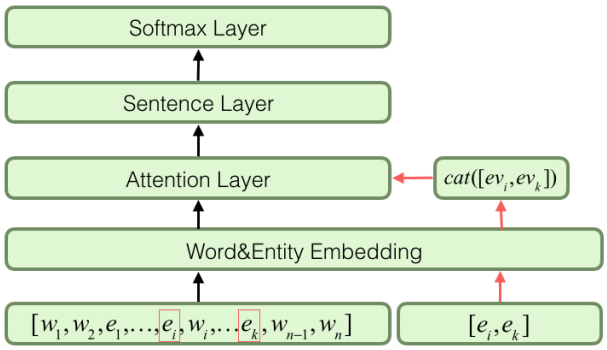

FIGURE 4. Attention deep network for relation extraction (RE).

1) EMBEDDING LAYER

The embedding layer is usually generated from Wikipedia and other text data by the word2vec algorithm [5]. The embedding layer using on the simulated data and the real data is not exactly the same. On simulation data experiment, embedding layer for common words and entity words both need to be trained by the deep learning network, therefore the difference between general words and entity words is only the index of different words. While on the real data experiment, because of the lack of large-scale training vectors, in order to make the general words embedding semantic, it is necessary to use the pre-trained embedding vectors. While the vector of the entity word still needs training. For the real data, the general words embedding matrix is different from the matrix of entity words, and only entity word embedding need training. Meanwhile, the embedding of named entities requires a special process. The mRE focuses on the relations between person, organization, and location. As a relation is contained mainly in the sentence meaning rather than in the specific entities, we uniformly embed all entity words in nine vectors, [*sPER*, *sORG*, *sLOC*, *oPER*, *oORG*, *oLOC*, *ePER*, *eORG*, *eLOC*]. The three vectors whose first letter is s (s*) indicate the entity words placed at E_1, and e* represents the entity words at E_2. The entities appearing in neither E_1 nor E_2 are in the sentence using o*. This embedding method reduces the impact of entity words.

2) ATTENTION LAYER

For different relation types, different words have different effects. When the entity pair is $\langle LOC, LOC \rangle$, the words that characterize the location, such as "capital" and "locate", are very important. For *PER* entities, however, the impact of these words is relatively weak. Therefore, the neural network structure for training needs to embody a mechanism by which the entity pair can influence the weight of other words in the sentence. In neural network structures, the attention model [21] can make use of the entity pairs to modify the weights of different words in a sentence. The attention model calculation process is as follows: First, obtain the word vectors $[w_i]$ in the sentence through the embedding layer. Then, concatenate the vectors of the entity pair with each word, $[e_1, e_2, w_i]$. The spliced vectors are passed to a fully connected layer, (formula (1)), and then the attention weight is obtained using the softmax function (formula (2)).

$$a_i = \tanh(W_{1\times L} \cdot [e_1, e_2, w_i]) \tag{1}$$

$$att = softmax([a_1, a_2, \ldots, a_n]) \tag{2}$$

Finally, multiply each word in the sentence by the corresponding weight $[att_1, \ldots, att_n]$. w_{a_i} is the vector that is modified by the entity pair $\langle e_1, e_2 \rangle$.

$$w_{a_i} = w_i \times att_i \tag{3}$$

3) SENTENCE EMBEDDING LAYER

After the processing of the embedding and attention layers, the sentence is expressed as $s = [w_{a1}, \ldots, w_{an}]$. Commonly used sentence vectorization methods are MLP, LSTM, and CNN [21]. Regardless of the order of the words, the sum of the MLP output can be taken as the sentence embedding. We use MLP as the sentence embedding layer in the simulation data (details are given in Section IV). In actual data, however, as the order of words in a sentence affects the sentence's meaning, CNN and LSTM are frequently used. During training, it is often the case that all samples are classified into NA because the number of NA samples is much greater than that for other categories. Once the samples are all predicted into NA, we reinitialize the network and iterate again. In our preliminary experiments, we found that

in contrast with LSTM, CNN does not tend to predict all samples into NA.

4) OUTPUT LAYER

In mRE, a single sentence with a pair of entities contains only one relation. Thus, the output layer uses a softmax layer. According to the discussion in Section IV-A, the iteration method described in Section IV-B can be used to optimize the UNLD.

V. EXPERIMENT

As the true labels of big data samples are difficult to obtain, the error rate for the labeled samples cannot be calculated accurately. Therefore, we used simulation data to ascertain the feasibility of the iteration process. Then, we used the proposed framework to build the training set for mRE.

A. SIMULATION DATA SET CONSTRUCTION

The simulation data set contains 11 categories. In real data, there are certain keywords in sentences that indicate a particular relation, such as ''location'' or ''capital'' for the LocatedIn relation. Based on this characteristic of language, we assume that each category has its special words. The size of word set W is 2501, of which words 0–1999 have no special meaning, and among words 2000–2500, each 50 words are the special words for a particular category. Here, we introduce the process for generating simulation samples. Each sentence in the data set except the NA samples is generated as follows:

(1) Choose five words randomly from the special words of a particular category.

(2) Choose fifteen words from word set W.

(3) Choose two special entity types, and combine all the words with entity types into one sample.

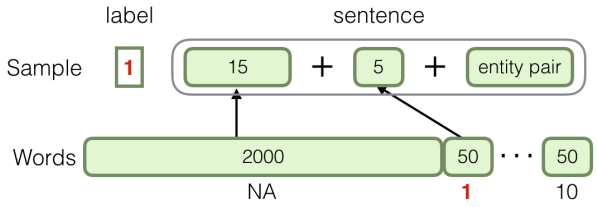

FIGURE 5. Example of sample generation.

Fig. 5 shows an example of a simulation sample that combines 15 words from words 0–1999 with 5 words from words 2000–2049 as the sentence. According to the generation process, each sentence contains 22 words, includes two entities, and is labeled as the category of the special words. (Our published code contains the specific details.) According to the above process, the true labels of the samples are known.

Now, we introduce the composition of the UNLD. The construction of each category in the UNLD is shown in Table 1. There are three parameters in the table: N represents the number of NA samples that are mixed into each category, O indicates the number of non-NA category samples that are

TABLE 1. Construction of UNLD set.

Category	Real	-	Other (O)
NA	100,000	-	5000*10
Category	**Real**	**NA (N)**	**non-NA (M)**
C1	8000	500	50*9
C2	7000	500	50*9
C3	6000	500	50*9
C4	5000	500	50*9
C5	4500	500	50*9
C6	4000	500	50*9
C7	3500	500	50*9
C8	3000	500	50*9
C9	2000	500	50*9
C10	1000	500	50*9

in the NA samples, and M is the number of other category samples mixed into one category. In Table 1, $N = 500$, $O = 5000$, and $M = 50$. For example, category C9 contains a total of 2950 samples, of which 2000 are actual C9 samples, 500 are NA samples, and 450 are other category samples (50 of each category). Parameters N and M are used to obfuscate the data. Setting N to 500 rather than to a larger value prevents incorrectly labeled samples from overwriting features of real samples. When $N = 300$ and $M = 50$, the error rate of the C9 samples is close to 30%, which is high enough for noise because a 30% error rate does not generally appear in actual labeled samples. The following sections primarily discuss the influence of parameter O.

B. EXPERIMENTAL RESULTS

1) ANALYSIS OF TEST RESULTS

In the experiment, the value of parameter O was selected from among the values 1000, 2000, 3000, 5000, 7000, and 8000. Fig. 6 shows the recall rate for each category on different O values after six iterations. The x-axis is the value of O, and the y-axis is the recall rate for the real labels. As shown in Fig. 6(a), the recall rate for categories with few real samples (e.g., C9 and C10) gradually decreased as O increased, which accords with our expectations. When the number of positive samples for an NA category is larger than the actual number of labeled samples, it is difficult for the classifier to accurately predict the category samples. If there are too many samples of any one NA category, some of the specific words of this category are treated as NA feature words.

Fig. 6(b) shows the number of samples transferred from NA to each category in each iteration when the value of O was 5000. For categories having a larger number of samples, the recall rate rose rapidly in the early iterations, as in C1–C5. If the number of error samples is more than twice that of the correct samples, then it is difficult to increase the recall rate with the iterations, as in C10. Let r denote the number of true labeled samples in one category, and let r_{na} represent the number of such samples in NA. When $r_{na}/2 < r$, the category achieves a higher recall rate by iterating. The precondition is that the labeled samples cover as many kinds of features of this category as possible, which means that it is necessary to include as many keywords as possible in the labeled samples

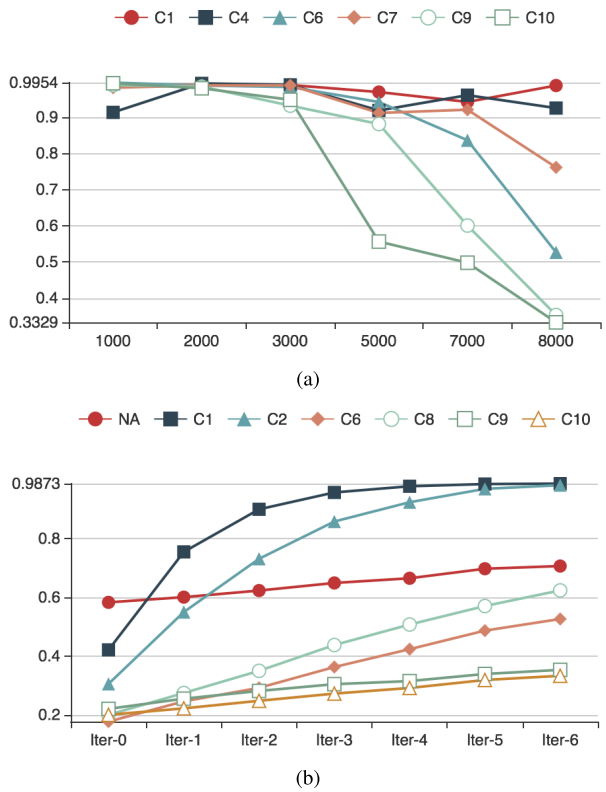

FIGURE 6. Experiment results for deep learning framework. (a) Variation in recall rate with *O*. (b) Change in recall at each iteration.

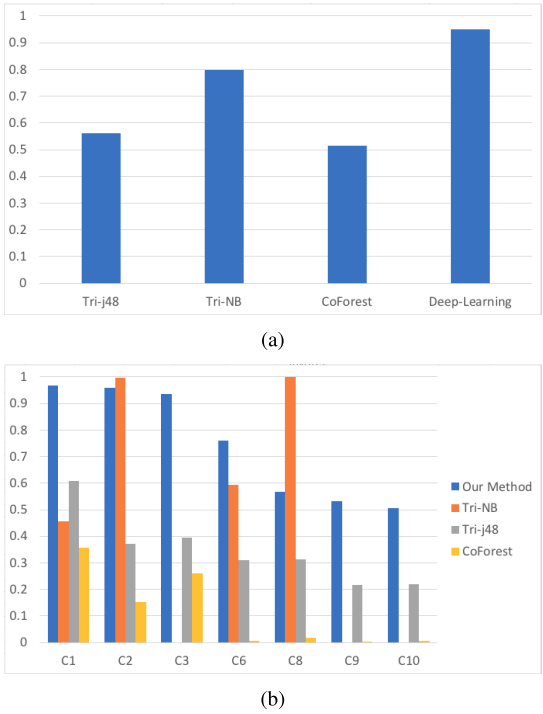

FIGURE 7. Comparison of various algorithms. (a) Precision of different algorithms. (b) F1 values for different categories.

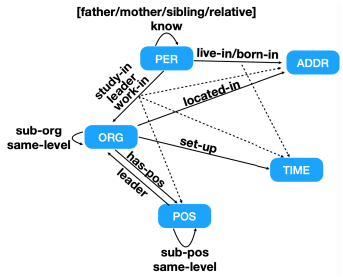

FIGURE 8. The relations in real data.

for the mRE problem. Meanwhile, the *AWN* value and the true accuracy of the non-NA categories are greater than 0.95 for the entire process. In order to increase the recall rates of $r_{na}/2 > r$, two methods are available: (1) Add unlabeled samples to NA in batches; (2) reduce the weight of the NA samples or increase the weights of small categories, such as C9 and C10. The first method entails higher training costs. However, when the number of unlabeled samples is very large, batch training methods must be used. The second method, modifying the weights of the samples, may reduce the accuracies of the labeled categories. After the final iteration, the hard sample should be checked to determine whether the auto-labeling is reasonable. At the same time, it is necessary to ensure than the number of real NA samples is at least twice as large as the number of non-NA samples. After the last iteration, the weights need to be removed and retrained, and then the non-NA samples predicted as NA should be checked.

We chose two commonly used semi-supervised algorithms, Tri-training [16] and Co-forest [22], to compare with our algorithm. The results are shown in Fig. 7. We used 20%–50% of the tag data for the semi-supervised learning algorithm training and then chose the best results for comparison. From the results, it can be seen that the proposed algorithm is superior to the traditional semi-supervised algorithms in terms of precision measures. Tri-NB has very random results for different categories, primarily because some special words in Bayesian modeling may have a pronounced effect on the prediction results. This effect is mainly due to noise in the training data, and traditional semi-supervised algorithms have no way to remove noise from the tag data. Thus, these incorrect tag data affect the iteration process in semi-supervised learning.

We used this framework to build the mRE training set. Thus far, we have collected 100,000 relation samples on Weibo and Baidupedia using our framework. The experiment consists of five type entities: Person, Organization, Position, Geographic location (Address), and Time. The relations between entities is shown in Fig. X. In addition to standard binary relations, Fig. X contains some n-ary relations, such as ''study-in'', ''work-in'' and ''live-in''. For example, the ''work-in'' represents a relation between Person and Organization, while a person can work in multiple companies, and a company also has multiple positions.

An example of ''work-in'' relation is shown in Fig. X. In the experiment, the n-ary relations needs to be transformed

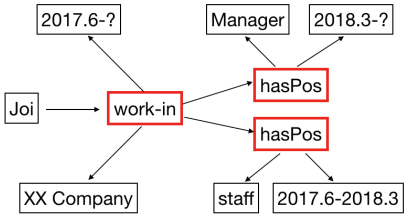

FIGURE 9. Example of n-ary relation "work-in".

into multiple binary relations. There are a large number of such binary relations in domain relation extraction, and there is no public training set available. In the experiment, 2000 initial samples are first labeled for each type of binary relation by the artificial and SnowBall. The labeling accuracy of the initial sample does not require rigorous review. Then iteratively using our algorithm. At present, the total number of labeled samples has exceeded 200,000, and the accuracy of manual sampling is above 90%, but the recall rate of some relation is about 70%, and there is still room for optimization.

VI. CONCLUSION

In this paper, we have proposed an efficient sample marking algorithm for mRE. Unlike the traditional semi-supervised methods, our method uses all samples for training from the beginning. In order for unlabeled samples to participate in training, we introduced the construction of a UNLD set and proposed an iterative optimization method for relabeling. Further, we analyzed the conditions for iterative labeling. Using deep learning, we translated the needs of feature engineering into the design of a network structure that is suitable for the RE problem. The experimental results show that our method can effectively improve the recall rates for non-NA categories in the UNLD and that for large categories, the recall rates are increased from 0.4 to nearly 1.

REFERENCES

[1] J. Jiang, "Information extraction from text," in *Mining Text Data*. Boston, MA, USA: Springer, 2012, pp. 11–41.
[2] N. Kambhatla, "Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations," in *Proc. ACL Interact. Poster Demonstration Sessions*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2004, Art. no. 22.
[3] R. J. Mooney and R. C. Bunescu, "Subsequence kernels for relation extraction," in *Proc. Adv. Neural Inf. Process. Syst.*, 2006, pp. 171–178.
[4] A. Madaan, A. Mittal, G. R. Mausam, G. Ramakrishnan, and S. Sarawagi, "Numerical relation extraction with minimal supervision," in *Proc. AAAI*, 2016, pp. 2764–2771.
[5] M. Zhang, Y. Zhang, and G. Fu, "End-to-end neural relation extraction with global optimization," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 1730–1740.
[6] 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.
[7] A. Sun, R. Grishman, and S. Sekine, "Semi-supervised relation extraction with large-scale word clustering," in *Proc. 49th Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2011, pp. 521–529.
[8] J. Chen, D. Ji, C. L. Tan, and Z. Niu, "Relation extraction using label propagation based semi-supervised learning," in *Proc. 21st Int. Conf. Comput. Linguistics 44th Annu. Meeting Assoc. Comput. Linguistics*, 2006, pp. 129–136.
[9] G. Erkan, A. Ozgur, and D. R. Radev, "Semi-supervised classification for extracting protein interaction sentences using dependency parsing," in *Proc. Joint Conf. Empirical Methods Natural Lang. Process. Comput. Natural Lang. Learn. (EMNLP-CoNLL)*, 2007, pp. 228–237.
[10] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, "Distant supervision for relation extraction without labeled data," in *Proc. Joint Conf. 47th Annu. Meeting ACL 4th Int. Joint Conf. Natural Lang. Process. (AFNLP)*, vol. 2, 2009, pp. 1003–1011.
[11] R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld, "Knowledge-based weak supervision for information extraction of overlapping relations," in *Proc. 49th Annu. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, 2011, pp. 541–550.
[12] M. Surdeanu, J. Tibshirani, R. Nallapati, and C. D. Manning, "Multi-instance multi-label learning for relation extraction," in *Proc. Joint Conf. Empirical Methods Natural Lang. Process. Comput. Natural Lang. Learn.*, 2012, pp. 455–465.
[13] S. Vashishth, R. Joshi, S. S. Prayaga, C. Bhattacharyya, and P. Talukdar. (2018). "RESIDE: Improving distantly-supervised neural relation extraction using side information." [Online]. Available: https://arxiv.org/abs/1812.04361
[14] S. Brin, "Extracting patterns and relations from the World Wide Web," in *Proc. Int. Workshop World Wide Web Databases*. Berlin, Germany: Springer, 1998, pp. 172–183.
[15] E. Agichtein and L. Gravano, "Snowball: Extracting relations from large plain-text collections," in *Proc. 5th ACM Conf. Digit. Libraries*, 2000, pp. 85–94.
[16] Z.-H. Zhou and M. Li, "Tri-training: Exploiting unlabeled data using three classifiers," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 11, pp. 1529–1541, Nov. 2005.
[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.
[18] H.-T. Cheng *et al.*, "Wide & deep learning for recommender systems," in *Proc. 1st Workshop Deep Learn. Recommender Syst.*, 2016, pp. 7–10.
[19] Z. Zhou, J. Shin, L. Zhang, S. Gurudu, M. Gotway, and J. Liang, "Fine-tuning convolutional neural networks for biomedical image analysis: Actively and incrementally," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, Jul. 2017, pp. 7340–7349.
[20] C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky, "The stanford CoreNLP natural language processing toolkit," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics, Syst. Demonstrations*, 2014, pp. 55–60.
[21] A. M. Rush, S. Chopra, and J. Weston. (2015). "A neural attention model for abstractive sentence summarization." [Online]. Available: https://arxiv.org/abs/1509.00685
[22] N. Settouti, M. El Habib Daho, M. El Amine Lazouni, and M. A. Chikh, "Random forest in semi-supervised learning (co-forest)," in *Proc. 8th Int. Workshop Syst., Signal Process. Appl. (WoSSPA)*, May 2013, pp. 326–329.

Authors' photographs and biographies not available at the time of publication.