DIAG-NRE: A Deep Pattern Diagnosis Framework for Distant Supervision Neural Relation Extraction

Shun Zheng\textsuperscript{1}, Peilin Yu\textsuperscript{2}, Lu Chen\textsuperscript{1}, Ling Huang\textsuperscript{1,3} and Wei Xu\textsuperscript{1}

\textsuperscript{1} Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China  
\textsuperscript{2} Department of Computer Sciences, The University of Wisconsin-Madison, Madison, USA  
\textsuperscript{3} AHI Fintech Inc., Beijing, China

\texttt{zhengs14@mails.tsinghua.edu.cn, peilin@cs.wisc.edu, lchen7@mails.tsinghua.edu.cn, huang.ling@gmail.com, weixu@tsinghua.edu.cn}

Abstract

Modern neural network models have achieved the state-of-the-art performance on relation extraction (RE) tasks. Although distant supervision (DS) can automatically generate training labels for RE, the effectiveness of DS highly depends on datasets and relation types, and sometimes it may introduce large labeling noises. In this paper, we propose a deep pattern diagnosis framework, DIAG-NRE, that aims to diagnose and improve neural relation extraction (NRE) models trained on DS-generated data. DIAG-NRE includes three stages: (1) The deep pattern extraction stage employs reinforcement learning to extract regular-expression-style patterns from NRE models. (2) The pattern refinement stage builds a pattern hierarchy to find the most representative patterns and lets human reviewers evaluate them quantitatively by annotating a certain number of pattern-matched examples. In this way, we minimize both the number of labels to annotate and the difficulty of writing heuristic patterns. (3) The weak label fusion stage fuses multiple weak label sources, including DS and refined patterns, to produce noise-reduced labels that can train a better NRE model. To demonstrate the broad applicability of DIAG-NRE, we use it to diagnose 14 relation types of two public datasets with one simple hyper-parameter configuration. We observe different noise behaviors and obtain significant F1 improvements on all relation types suffering from large labeling noises.

Introduction

Relation extraction (RE) is a vital task in natural language processing (NLP) and plays a key role in the knowledge base population (KBP) that can transform human-understandable texts into machine-understandable knowledge. The main goal of RE is to extract triplets from plain texts, such as transforming the sentence “Obama was born in Honolulu” with a head entity Obama and a tail entity Honolulu to a triplet (Obama, BornIn, Honolulu). The triplet, also referred to as the fact, is the basic factor of the knowledge base (KB).

A commonly adopted approach models RE as a supervised classification task that predicts the semantic relationship between two entities by the sentence meaning, such as (Zelenko, Aone, and Richardella 2003; Zhou et al. 2005). Recent progress in supervised RE includes neural-network-based relation extraction (NRE) models (Zeng et al. 2014; dos Santos, Xiang, and Zhou 2015; Wang et al. 2016; Zhou et al. 2016) that exhibit superior performance over traditional models based on handcrafted features.

However, NRE models require a large number of relation-specific human-annotated data to train, which are both expensive and time-consuming to collect. Instead, (Craven, Kumlien, and others 1999; Mintz et al. 2009) proposed distant supervision (DS) to automatically generate large-scale training data for RE. The main idea of DS is to align relational facts in KB to plain texts with entities detected in advance and roughly assign relation types from KB to the label of the corresponding matched texts. The assumption of DS is that if two entities hold a relation in KB, every sentence that contains these two entities describes this relationship. Although DS is both simple and useful in many cases, it also introduces intolerable labeling noises in others, as (Riedel, Yao, and McCallum 2010) found that the noisy-labeling problem became severe when the KB and the text corpus did not match well. The wrong labels contain: false positives (not all sentences that mention two entities with relations stored in KB express these relations) and false negatives (some sentences do describe two entities about a target relation, but this fact has not been covered by KB yet).

There are three categories of attempts to tackle the noisy-labeling problem to improve DS.

The first is to design specific model architectures that can better tolerate labeling noises, such as the multi-instance learning paradigm (Riedel, Yao, and McCallum 2010; Hoffmann et al. 2011; Surdeanu et al. 2012; Zeng et al. 2015; Lin et al. 2016). These models relax the raw assumption of DS by grouping multiple sentences that mention the same entity pair together as a bag and assuming that at least one sentence in this bag expresses the relation. This weaker assumption can alleviate the noisy-labeling problem to some extent, but this problem still exists at the bag level, and (Feng et al. 2018) discovered that the bag-level model struggles to carry out sentence-level prediction.

The second attempts to automatically reduce labeling noises to produce cleaned labels. For example, (Feng et al. 2018) and (Qin, Xu, and Wang 2018) both adopted reinforcement learning (RL) to train an agent that interacts with an NRE model to learn how to remove or redistribute noisy labels. Although these methods work automatically without human intervention, a major limitation is that merely getting rewards from the matching degree between the DS-
generated label and the NRE model prediction is not enough because it cannot discover error labels that coincide with the model predictions.

The third seeks to add a few extra human efforts. \cite{Zhang et al. 2012, Pershina et al. 2014, Angeli et al. 2014, Liu et al. 2016} mixed a small set of crowd-annotated labels with purely DS-generated labels, but only sufficient large and high-quality labels can contribute significant improvements. Data programming (DP) \cite{Ratner et al. 2016} proposed a generative model to fuse weak labels from multiple sources, called labeling function (including DS-based and pattern-based heuristics), and then infer the true label distribution, but it required domain experts to produce relation-specific patterns. Although it seems quicker to write patterns rather than annotate plenty of examples, this process requires high-level skills since a pattern is essentially a small program and a human typically needs to examine many examples to write a good pattern. For example, the spouse relation in the DP example uses 11 functions with over 20 relation-specific keywords \cite{1}.

In this work, we propose a deep pattern diagnosis framework for NRE models, DIAG-NRE, to fill the gap between DS and DP. First, we build a pattern extraction agent by RL to generate relation-specific patterns through interaction with an NRE model trained on DS-generated data. Then, we build a pattern hierarchy to find the most representative patterns and let human reviewers evaluate them. For each pattern, we evaluate it in a quantitative way by annotating a certain number of pattern-matched examples. In this way, we minimize both the workload and the difficulty of human reviewers. After this pattern refinement stage, we obtain some positive patterns that highly infer the target relation and negative patterns that are not relevant to the relation. At last, by fusing weak labels generated by DS and refined patterns, we estimate the true label distribution as DP does and retrain a better NRE model.

In summary, our contributions include the following:

- We propose DIAG-NRE to diagnose and improve NRE models trained on DS-generated data. DIAG-NRE bridges the gap between DS and DP by both reducing the number of high-quality labels (v.s. more supervised approaches) and making the human work more comfortable (v.s. writing patterns).

- Our pattern extraction agent can generate patterns that highly motivate the NRE model to make the target relation prediction. These patterns not only help us to diagnose the labeling noises but also interpret hidden features learned by NRE models.

- We conduct extensive experiments on 14 relations and observe different noise behaviors. With one simple hyper-parameter configuration, DIAG-NRE achieves better F1 scores on 10 relation types over the best baseline. For six relations with large noise problems, DIAG-NRE can obtain the F1 improvement of over 5.0 points. Particularly, for one relation with severe labeling noises, we obtain an F1 improvement of over 40.0 points.

Related Work

With the rise of deep learning \cite{Bengio and others 2009}, multiple NRE architectures have been developed, including convolutional neural network (CNN) variants \cite{Zeng et al. 2014, Zeng et al. 2015, Wang et al. 2016} and long short-term memory (LSTM) network variants \cite{Xu et al. 2015, Miwa and Bansal 2016, Zhou et al. 2016}. However, the main bottleneck to quickly deploy NRE models in practice lies in the lack of labeled training data. As we have discussed in the introduction, DS can automatically generate large noisy training labels and there are several different attempts to alleviate the noisy-labeling problem.

Similar to DS-based extraction, Open Information Extraction (OpenIE) \cite{Banko et al. 2007, Mausam et al. 2012, Del Corro and Gemulla 2013, Angeli, Johnson Premkumar, and Manning 2015, Stanovsky and Dagan 2016, Cui, Wei, and Zhou 2018} is another extraction paradigm with no human intervention, but triplets extracted by OpenIE need to be further mapped into a target ontology.

As for the pattern extraction stage, we note that some existing methods adopt similar strategies as our pattern extraction agent to interpret neural networks for NLP tasks but with different purposes. For example, \cite{Zhang, Huang, and Zhao 2018} employed RL to interact with LSTM \cite{Hochreiter and Schmidhuber 1997} to find structured representations and thus improve the classification performance, and \cite{Li, Monroe, and Jurafsky 2016} used RL to find the decision-changing phrases to interpret neural networks on common NLP tasks, such as sentiment classification. However, NRE models are unique because we only care about the semantic relation of a given entity pair mentioned in the sentence. To the best of our knowledge, DIAG-NRE is the first work to employ RL to interpret NRE models.

We also note that the relational-pattern mining task has been extensively studied \cite{Califf and Mooney 1999, Carlson et al. 2010, Nakashole, Weikum, and Suchanek 2012, Jiang et al. 2017}. Different from those studies, our pattern generation process is simply based on RL, does not rely on NLP annotation tools (part-of-speech tagging, dependency parsing, etc) and establishes a link with the prediction of NRE models. Furthermore, extracted patterns aim to serve the weak-label-fusion model instead of performing pattern-based relation prediction in traditional studies. Besides, another relevant method is \cite{Takamatsu, Sato, and Nakagawa 2012} that infers negative patterns from the example-pattern relation co-occurrence and removes the wrong labels accordingly. In contrast, our framework is based on modern NRE models and not only reduces negative patterns but also reinforces positive patterns.

In this paper, we focus to bridge the gap between DS and DP \cite{Ratner et al. 2016}, a basic weak-label-fusion method. Designing a better generic weak-label-fusion mechanism is another hot research topic \cite{Varma et al. 2016, Bach et al. 2017, Liu et al. 2017}.
Methodology

DIAG-NRE contains three key stages: deep pattern extraction, pattern refinement and weak label fusion. In this section, we first formulate the common NRE models and then introduce these three stages separately. Figure 1 presents an overview of DIAG-NRE.

NRE Models

Although many NRE models with different network architectures have been developed, these models often share a common input-output schema. Given an instance with T tokens, NRE models get a sequence of dense word vectors, \( w = [w_1, w_2, \ldots, w_T] \in \mathbb{R}^{d^w \times T} \), by looking up a word embedding table, \( V \in \mathbb{R}^{d^v \times |V|} \), where \( d^w \) and \( |V| \) denote the word vector size and the total vocabulary size, respectively. Particularly, to enable a neural network’s awareness of the explicit entity pair in the sentence, we need to add extra position features. A popular approach is position embedding (Zeng et al. 2014). Its basic idea is to encode the relative distance between a word and each entity by a position embedding table \( P \in \mathbb{R}^{d^p \times |P|} \), where \( d^p \) denotes the position vector size and \( |P| \) denotes the total number of relative distances. Since there is a pair of entities for one sentence, we typically look up two embedding tables and end up with two sequences of position vectors, \( p^1 = [p^1_1, p^1_2, \ldots, p^1_T] \) and \( p^2 = [p^2_1, p^2_2, \ldots, p^2_T] \), where each \( p^i_j \in \mathbb{R}^{d^p} \). By concatenating the sequence of word vectors and position vectors, we obtain the final input representation of the instance \( s \) as \( x = [x_1, x_2, \ldots, x_T] \), where \( x_i = [w_i; p^1_i; p^2_i] \in \mathbb{R}^{d^w + d^p} \). Assuming our interested relation class is \( r \), NRE models perform different types of tensor manipulations on \( x \) and obtain the probability of \( r \) given the instance \( s \) as \( P_o(r|x) \), where \( o \) denotes NRE model parameters except for the input embedding tables.

STAGE 1: Deep Pattern Extraction

In this stage, we build an agent using RL to distill relation-specific patterns from NRE models. Our pattern extraction agent is compatible with any NRE model that shares the input-output schema we describe above.

Action. The agent takes an action, retaining or erasing, for each token in the instance \( s \) and transforms the input representation from \( x \) to \( \hat{x} \). During this process, the column \( i \) of \( x \), \( x_i = [w_i; p^1_i; p^2_i] \), corresponding to the token \( i \) of raw instance \( s \), is transformed into \( \hat{x}_i = [\hat{w}_i; \hat{p}^1_i; \hat{p}^2_i] \), where the position vectors are left untouched and the new word vector \( \hat{w}_i \) is adjusted based on the action taken by the agent. If the action is retaining, we set \( \hat{w}_i \) to be equal to the raw word vector \( w_i \). Otherwise, if the action is erasing, we set \( \hat{w}_i \) to be zero to remove the semantic meaning of the token \( i \). Our action design is aimed at distilling important tokens while keeping the position awareness. After taking a sequence of actions, \( a = [a_1; a_2; \ldots; a_T] \), where \( a_i \in \{0, 1\} \) (0: retaining, 1: erasing), we get the transformed representation \( \tilde{x} \) with \( \hat{T} \) tokens retained. Our action space is similar to the one used in (Li, Monroe, and Jurafsky 2016) [Zhang, Huang, and Zhao 2018] except that we keep the position information untouched.

Reward. Our purpose is to find the most simplified sequence that both enjoys the sparsity and preserves the raw prediction confidence. Thus, given the raw input representation \( x \) and the corresponding action vector \( a \), we define the reward as follows:

\[
R(a|x) = \log P_o(r|\tilde{x}) - \log P_o(r|x) + \eta \cdot (1 - \hat{T}/T),
\]

where the total reward is composed of two parts: one is the log-likelihood term to pursue high prediction confidence, and the other is the sparse ratio term to induce sparsity in terms of retained tokens. We balance these two parts through a hyper-parameter \( \eta \). This reward is similar to the one used in (Zhang, Huang, and Zhao 2018) to discover information-distilled structures for LSTM.

State. We design the state at the instance level that decouples the dependence among actions for one instance to better utilize the parallel computing resources and thus speed up the training process. Meanwhile, the state should also be independent of NRE model architectures. Actually, our agent only employs the input representation \( x \) obtained from the NRE model as the state.

Agent. We employ policy-based RL to train a neural network that can predict a sequence of actions for an instance to maximize the reward. As described in the state part, we decouple the dependence of predicting the action for each token. Our policy network estimates \( \pi_o(a|x) = \prod_{i=1}^T \pi_o(a_i|x) \) directly, and can calculate \( \pi_o(a_i|x) \) in parallel, where \( o \) denotes parameters of the policy network.

To enrich the context information to better predict the action for each token, our network employs the forward and backward LSTM networks to encode \( x \) into \( h \) as

\[
\vec{h} = [\vec{h}_1, \vec{h}_2, \ldots, \vec{h}_T] = \text{Forward-LSTM}(x),
\]

\[
\vec{\bar{h}} = [\vec{\bar{h}}_1, \vec{\bar{h}}_2, \ldots, \vec{\bar{h}}_T] = \text{Backward-LSTM}(x),
\]

\[
h = [h_1, h_2, \ldots, h_T] = \text{Concatenate}(\vec{h}, \vec{\bar{h}}),
\]

where \( \vec{h}_i \in \mathbb{R}^{d^h}, \bar{h}_i \in \mathbb{R}^{d^h}, h_i = [\vec{h}_i; \bar{h}_i] \in \mathbb{R}^{2d^h} \), and \( d^h \) denotes the size of LSTM’s hidden state.

Then, we employ an attention-based strategy (Bahdanau, Cho, and Bengio 2014) to aggregate the context information as \( c = [c_1, c_2, \ldots, c_T] \). For each token \( i \), we compute the context vector \( c_i \in \mathbb{R}^{2d^h} \) as follows:

\[
c_i = \sum_{j=1}^T \alpha^j_i h_j,
\]

where each scalar weight \( \alpha^j_i \) is calculated by \( e^j_i / (\sum_{k=1}^T e^k_i) \), and \( e^j_i \) is computed by a small network \( q(x_i, h_j) \) as

\[
q(x_i, h_j) = v^q_k \tanh(W^q x_i + W^h h_j),
\]

In this work, we refer to a sentence together with two paired entities as an instance and omit the instance index for brevity.
Deep Pattern Extraction

Pattern Refinement

Weak Label Fusion

Figure 1: An overview of DIAG-NRE with three stages.

where \( W_x \in \mathbb{R}^{2d_h \times d_z}, W_h \in \mathbb{R}^{2d_h \times 2d_h} \) and \( v_o \in \mathbb{R}^{2d_h} \) are network parameters.

Next, we compute the final representation to infer actions as \( z = [z_1, z_2, \cdots, z_T] \), and for each token \( i \), \( z_i = [x_i; c_i] \in \mathbb{R}^{d_z + 2d_h} \) incorporates word, position and context information.

Finally, we estimate the probability of taking action \( a_i \) for token \( i \) as

\[
\pi_\Theta(a_i|x) = o_i^{a_i} \cdot (1 - o_i)^{(1-a_i)},
\]

where \( o_i = \text{sigmoid}(W_o^T z_i + b_o) \), \( W_o \in \mathbb{R}^{d_z + 2d_h} \) and \( b_o \in \mathbb{R} \) are network parameters.

Optimization. We employ the REINFORCE algorithm (Williams 1992) and policy gradient methods (Sutton et al. 2000) to optimize parameters of the policy network described above, where the basic idea is to rewrite the gradient formulation and then apply the back-propagation algorithm (Rumelhart, Hinton, and Williams 1986) to update network parameters. We define our objective as

\[
L(\Theta) = \mathbb{E}_x \left[ E_{\pi_\Theta(a|x)}R(a|x) \right].
\]

By taking the derivative of \( J(\Theta) \), we can obtain the following gradient formulation:

\[
\nabla_\Theta L(\Theta) = \mathbb{E}_x \left[ E_{\pi_\Theta(a|x)} \nabla_\Theta \log \pi_\Theta(a|x) \right].
\]

Based on the above formulation, we can automatically obtain approximate gradients by combining the likelihood ratio trick with the auto differentiation functionality of modern deep learning packages.

After the training process, the agent has learned to retain relation-specific tokens and we can move on to the next stage.

STAGE 2: Pattern Refinement

By evaluating the trained agent on training instances, we collect plenty of agent actions. In this section, we show how to induce patterns from these actions and associated instances.

Pattern Induction. For each instance, we retain two entity tokens with their entity types and all other tokens that the agent chooses to retain. To incorporate the position information, we divide the relative distance between two adjacent retained tokens into four categories: consecutive (no tokens between them), short (1-3 tokens), medium (4-9 tokens) and long (10 or more tokens) distance. Patterns in this format can encode several important information, such as entity types, key tokens and relative distances between retained tokens. For example, given a sentence “Joachim Fest was born in Berlin”, the head entity Joachim Fest with the type PERSON and the tail entity Berlin with the type CITY, assuming the agent decides to retain tokens with the index in \{0, 2, 3, 4\}, we conclude a pattern as "ENTITY1:PERSON:PAD1,3:born in ENTITY2:CITY:in which ENTITY1:PERSON and ENTITY2:CITY denotes the head and the tail entities with required entity types, respectively, and PAD{1,3} denotes the allowed frequency range of any other token at its position.

Pattern Hierarchy. We merge induced patterns by grouping multiple instances with the same pattern and recording the pattern source count that denotes the number of instances producing this pattern, because we observe that it is a crucial metric to measure the representativeness of the pattern. Then, we build a hierarchy for merged patterns according to pattern-matched instances. In this hierarchy, the parent pattern should cover all instances matched by the child pattern, and any pattern without parents lies in the first level. To speed up the hierarchy construction process, we only consider patterns with the source count larger than \( n_s \).

Afterward, we traverse the first level of the pattern hierarchy and follow the decreasing order of the pattern source count to select top \( n_h \) most representative patterns. To quantitatively examine the pattern quality, we adopt an approximation method by randomly selecting \( n_o \) pattern-matched instances and annotating them manually. Thus, for each relation type, we end up with \( n_s, n_h, n_o, p_h \) hand-tagged instances. We assign patterns with the accuracy higher than \( p_h \) and lower than \( p_l \) into the positive pattern set and the negative pattern set, respectively, to serve the next stage, where \( n_s, n_h, n_o, p_h \) and \( p_l \) are hyper-parameters at the current stage. In our experiments, we show that one simple configuration can adapt to all 14 relation types.

STAGE 3: Weak Label Fusion

DP (Ratner et al. 2016) proposed an abstraction of the weak label generator, called the labeling function (LF), that can
incorporate both DS and heuristic patterns. Considering a binary classification task, an LF outputs one label (+1: positive, -1: negative or 0: unknown) for each input instance. In our case, the LF of DS generates +1 or -1, LFs of positive patterns generate +1 or 0, and LFs of negative patterns generate -1 or 0. We also use the basic generative model described in (Ratner et al. 2016). Assuming we have m labeling functions and the prior of each class is 0.5, we can estimate \( P_{\alpha,\beta}(L^s, Y^s) \), as
\[
\frac{1}{2} \prod_{i=1}^{m} (\beta_i \alpha_i 1_{L_i^s = Y^s} + \beta_i (1 - \alpha_i) 1_{L_i^s = -Y^s}) + (1 - \beta_i) 1_{L_i^s = 0},
\]
where \( L_i^s \in \{-1, 0, +1\} \) denotes the weak label generated for instance \( s \) by the \( i \)-th labeling function, and \( \alpha \) and \( \beta \) are model parameters we need to estimate. According to (Ratner et al. 2016), unsupervised parameter estimation on an unlabeled instance set \( S \) by solving \( \max_{\alpha,\beta} \sum_{s \in S} \log P_{\alpha,\beta}(L^s, Y^s) \) needs some strong assumptions to be correct.

However, in our case, we obtain a small labeled set \( S_L \) at the pattern refinement stage. So, we use this labeled set to estimate \( (\alpha, \beta) \) by solving
\[
\max_{\alpha,\beta} \sum_{s \in S_L} \log P_{\alpha,\beta}(L^s, Y^s),
\]
where the closed-form solutions are
\[
\alpha_i = \frac{\sum_{s \in S_L} 1_{L_i^s = Y^s}}{\sum_{s \in S_L} 1_{L_i^s = Y^s} + 1_{L_i^s = -Y^s}},
\]
\[
\beta_i = \frac{\sum_{s \in S_L} 1_{L_i^s = -Y^s}}{|S_L|}
\]
for each \( i \in \{1, \cdots, m\} \). After estimating these parameters, we can infer the true label distribution by the posterior \( P_{\alpha,\beta}(Y^s|L^s) \) and use the soft label to train a better NRE model, as (Ratner et al. 2016) does.

**Experiments**

Since different relation types/datasets have different noise behaviors when applying the DS strategy, we perform experiments on multiple relations one by one to present corresponding diagnosis effects clearly. Specifically, for each relation, we solve a binary classification task that distinguishes the expression of a sentence about two entities to be the target relation or unknown (denoted as \( \text{NA} \)). Since our purpose is to diagnose and reduce labeling noises caused by DS, we only evaluate the NRE model at the sentence level in this work, as (Ratner et al. 2016) does.

**Experimental Setup**

**Data.** To study the DS-caused noises on different relation types, we select 10 relations from the NYT dataset (NYT), first presented in (Riedel, Yao, and McCallum 2010), with

\[ \text{http://lesl.cs.umass.edu/riedel/ecml/} \]

| TID | Relation Abbreviation | Train | Test |
|-----|-----------------------|-------|------|
| R0  | /bus./per./company     | 5.3k  | 186  |
| R1  | /loc./gov./administra. | 4.9k  | 180  |
| R2  | /loc./gov./capital     | 5.3k  | 20   |
| R3  | /loc./gov./contains    | 44.6k | 263  |
| R4  | /loc./adm./country     | 4.9k  | 89   |
| R5  | /loc./hei./neighborhood| 5.6k  | 55   |
| R6  | /peo./per./nationality | 7.5k  | 84   |
| R7  | /peo./per./place_lived | 6.7k  | 230  |
| R8  | /peo./per./place_of_birth | 3.1k | 16   |
| R9  | /peo./dec./place_of_death | 1.9k | 19   |
| R10 | /peo./dec./place_of_death | 5.7k | 1.3k |

**Table 1:** The total 14 RE tasks with corresponding task IDs (TIDs), relation abbreviations and positive label counts in the training and testing sets. The training set, generated by DS, contains 453,224 and 395,739 instances for NYT and UW, respectively. The testing set, annotated by the human, contains 1,028 and 15,623 instances for NYT and UW, respectively.

This table shows how enough DS-generated labels as (Qin, Xu, and Wang 2018) does and all four relations from the UW dataset (UW), developed by (Liu et al. 2016). NYT contains a training set and a testing set both created by DS with 522,611 and 172,448 sentences, respectively. UW contains a training set created by DS, a crowd-annotated set and a minimal hand-tagged testing set with 676,882, 18,128 and 164 sentences, respectively. Then, we extract from the raw corpus for each relation to construct a binary classification task. Note that we include any instance with multiple labels only once. Besides, we re-annotate the raw sentence by the Stanford CoreNLP 3.9.1 to get entity types.

**Evaluation & Ground Truth.** To accurately evaluate the model performance, we adopt the manual evaluation that uses ground truth labels as (Ratner et al. 2016); Liu et al. 2016 does. We do not use the heldout evaluation (Minz et al. 2009), because it inherently contains many noises and cannot demonstrate the noise reduction effect clearly.

For the NYT dataset, we randomly select up to 100 instances per relation (including the special unknown relation \( \text{NA} \)) from the testing set and manually annotate them. We obtain 1,028 hand-tagged instances as the ground-truth for evaluation.

For the UW dataset, the raw testing set is too small, but the crowd-annotated set has broad coverage and very high quality (88% agreement with hand-tagged labels according to (Liu et al. 2016)), and does not overlap with the training set; thus, we use the crowd-annotated set as the ground truth.

Table 1 summarizes the details of 14 RE tasks.

For more details on the data and instructions, please refer to

\[ \text{https://www.cs.washington.edu/ai/gated_instructions/naacl_data.zip} \]

\[ \text{https://stanfordnlp.github.io/CoreNLP/} \]
### Table 2: Total comparison results for 14 tasks with three baselines, where we report three metrics: precision (P), recall (R) and f1 score (F1) for each method, indicate the increment of F1 with our method over the vanilla DS (Inc-DS) and the best baseline (Inc-Best), and highlight the best F1 and any F1 increment that is higher than 5.0.

| TID | Distant Supervision | Gold Label Mix | RLRE | DIAG-NRE |
|-----|---------------------|----------------|------|---------|
|     | P       | R       | F1   | P       | R       | F1   | Inc-DS | Inc-Best |
| R₀  | 95.1    | 41.5    | 57.8 | 95.7    | 40.8    | 57.2 | 97.7   | 32.4   | 48.6   | 95.7   | 42.8   | 59.1   | +1.4   | +1.4   |
| R₁  | 91.9    | 9.1     | 16.4 | 90.2    | 11.7    | 20.2 | 92.6   | 4.2    | 8.0    | 94.5   | 44.8   | 60.7   | +44.3  | +40.4  |
| R₂  | 37.0    | 83.0    | 50.8 | 40.0    | 85.0    | 54.0 | 64.8   | 68.0   | 66.1   | 42.4   | 85.0   | 56.0   | +5.2   | -10.1  |
| R₃  | 87.5    | 79.2    | 83.2 | 87.1    | 80.2    | 83.5 | 87.5   | 79.2   | 83.2   | 87.0   | 79.8   | 83.2   | +0.0   | -0.3   |
| R₄  | 95.3    | 50.1    | 64.7 | 94.1    | 49.0    | 63.9 | 98.2   | 47.9   | 64.0   | 94.5   | 57.5   | 71.5   | +6.7   | +6.7   |
| R₅  | 82.7    | 29.1    | 42.9 | 84.7    | 29.5    | 43.6 | 82.7   | 29.1   | 42.9   | 84.5   | 37.5   | 51.8   | +8.9   | +8.3   |
| R₆  | 82.0    | 83.8    | 82.8 | 81.6    | 84.0    | 82.7 | 82.0   | 83.8   | 82.8   | 81.5   | 83.3   | 82.3   | -0.5   | -0.5   |
| R₇  | 82.3    | 22.3    | 35.1 | 82.0    | 22.6    | 35.4 | 83.5   | 21.8   | 34.5   | 82.0   | 25.6   | 39.0   | +3.8   | +3.6   |
| R₈  | 66.2    | 32.5    | 39.8 | 70.5    | 47.5    | 55.8 | 66.2   | 32.5   | 39.8   | 73.4   | 61.3   | 65.5   | +25.7  | +9.7   |
| R₉  | 85.4    | 73.7    | 77.9 | 85.9    | 80.0    | 81.5 | 85.4   | 73.7   | 77.9   | 89.0   | 87.4   | 87.1   | +9.2   | +5.6   |
| R₁₀ | 35.9    | 75.7    | 48.7 | 35.8    | 75.0    | 48.5 | 36.0   | 75.3   | 48.7   | 36.2   | 74.5   | 48.7   | +0.0   | -0.0   |
| R₁₁ | 57.8    | 18.5    | 28.0 | 59.3    | 19.1    | 28.8 | 57.8   | 18.5   | 28.0   | 56.3   | 23.5   | 33.1   | +5.1   | +4.3   |
| R₁₂ | 37.3    | 64.0    | 46.9 | 40.0    | 64.9    | 49.1 | 37.3   | 64.0   | 46.9   | 48.1   | 71.9   | 57.5   | +10.6  | +8.3   |
| R₁₃ | 77.1    | 71.3    | 74.0 | 77.5    | 70.3    | 73.5 | 77.1   | 71.3   | 74.0   | 80.7   | 71.1   | 75.4   | +1.5   | +1.5   |

### NRE Configuration.
For the NRE model, we implement a simple yet effective LSTM-based architecture described in (Zhou et al. 2016). We use the same set of hyper-parameters that can produce good results in all 14 tasks: word vectors ($d_w = 100$) are initialized by Glove vectors (Pennington, Socher, and Manning 2014); we set the position vector size $d_p$ to 5, the LSTM hidden size to 200, and the dropout probability at the embedding layer, the LSTM layer and the last layer to 0.3, 0.3 and 0.5, respectively; the optimizer is Adam (Kingma and Ba 2014) with a learning rate of 0.001, and the batch size is 5. Besides, we decide the early stopping epoch with cross-validation on each task independently. We implement neural models based on PyTorch and directly use its default parameter initialization strategy.

### Diagnosis Configuration.
We use a single diagnosis configuration for all 14 tasks. For our RL agent, the LSTM hidden size is 200, the optimizer is Adam with a learning rate of 0.001, the batch size is 5, and the training epoch is 10. In the deep pattern extraction stage, we alter the hyper-parameter $\eta$ in $\{0.1, 0.5, 1.0, 1.5\}$ to train multiple agents that tend to squeeze patterns with different granularities. To speed up the agent training and avoid unnecessary patterns, we only take the top 10,000 instances with the highest prediction probabilities of the target relation. At the pattern refinement stage, hyper-parameters include $n_s = 3$, $n_r = 20$, $n_a = 10$, $p_h = 0.8$ and $p_l = 0.1$. Thus, we get 200 hand-tagged instances (about 0.05% of the entire training set) and at most 20 patterns for the weak label fusion stage for each task.

Next, we first present the overall performance comparisons and then show how DIAG-NRE benefits from the diagnosis results.

### Table 3: Diagnosis results for each task, where columns represent the precision, recall and accuracy of DS-generated labels measured on 200 hand-tagged labels, as well as the number of positive and negative patterns preserved after the pattern refinement.

| TID | Prec. | Recall | Acc. | #Pos. | #Neg. |
|-----|-------|--------|------|-------|-------|
| R₀  | 100.0 | 81.8   | 82.0 | 20    | 0     |
| R₁  | 93.9  | 33.5   | 36.2 | 18    | 0     |
| R₂  | 75.7  | 88.0   | 76.5 | 9     | 5     |
| R₃  | 100.0 | 91.4   | 92.0 | 20    | 0     |
| R₄  | 93.3  | 72.4   | 80.9 | 10    | 2     |
| R₅  | 93.8  | 77.3   | 86.5 | 15    | 0     |
| R₆  | 88.3  | 76.9   | 75.1 | 14    | 0     |
| R₇  | 91.9  | 64.6   | 64.0 | 20    | 0     |
| R₈  | 29.3  | 30.4   | 30.4 | 60    | 4     |
| R₉  | 66.7  | 38.1   | 74.4 | 6     | 11    |
| R₁₀ | 81.8  | 90.7   | 81.0 | 7     | 0     |
| R₁₁ | 93.5  | 70.7   | 68.3 | 17    | 1     |
| R₁₂ | 35.0  | 70.0   | 60.0 | 4     | 15    |
| R₁₃ | 87.5  | 59.2   | 67.7 | 12    | 5     |

### Performance Comparisons
Based on the above configuration, DIAG-NRE can produce noise-reduced labels to retrain a better NRE model. In this part, we present the impact of different types of training labels on the final model performance.

### Baselines
We adopt the following baselines:
- **Distant Supervision** denotes the strategy developed by (Craven, Kumlien, and others 1999; Mintz et al. 2009).
- **Gold Label Mix**, studied in (Liu et al. 2016), mixes human-annotated high-quality labels with noisy labels generated by DS. In our case, we use the same 200 in-
Table 3, where we highlight metrics in which DS performs.

Table 4: Positive (Pos.), negative (Neg.) patterns and their matched examples with labels produced by different methods. For RLRE, None means the instance is removed. For DIAG-NRE, we present the soft label $P_{a,d}(Y^s = 1|L_s)$. For each pattern, we present DS Label as the number of DS-generated positive labels over the number of pattern-matched instances.

- **RLRE**, developed in (Feng et al. 2018), attempts to train an RL-based agent that can select correct-labeled instances by only interacting with the NRE model trained on DS-generated data. In our case, we follow the implementation of (Feng et al. 2018) to produce selected instances into a new training set for each task except that we use the LSTM-based NRE model.

To focus on the effect of labels produced with different methods, we fix all other variables for each task. Since the initialization of neural networks can also have large influences on the generalization performance, we run each NRE model with five fixed random seeds, ranging from 0 to 4, for each method on each task, and present the average metric in Table 2.

For a majority of tasks, including $R_1$, $R_4$, $R_5$, $R_8$, $R_9$ and $R_{10}$, we obtain the F1 improvement of over 5.0 points compared with the best baseline. Notably, the F1 improvement of the task $R_1$ has reached 40 points. For some tasks that DS generates a few noises, including $R_0$, $R_7$, $R_2$ and $R_{10}$, our method can obtain small improvements. For a few tasks, such as $R_3$, $R_6$ and $R_{10}$, only using DS is sufficient to train competitive models, and fusing other weak labels may have negative effects, but the negative impact is small.

Another interesting observation is that RLRE yields the best result on tasks $R_2$ and $R_{10}$ but gets worse results than the vanilla DS on tasks $R_0$, $R_1$, $R_4$ and $R_7$. Since the instance selector used in RLRE is hard to interpret, we can hardly figure out the specific reason. We conjecture that this behavior is due to the gap between maximizing the likelihood of the NRE model and the right instance selection.

**Diagnosis Results**

The deep pattern extraction and the pattern refinement stages not only provide refined patterns for the weak label fusion stage but also own the ability to interpret different noise effects of DS-generated labels. We present diagnosis results in Table 3, where we highlight metrics in which DS performs poorly. We illustrate the noise effects of DS-generated labels from two perspectives: false negatives and false positives.

- **False Negatives**: A typical example is task $R_1$ (/loc./cou./administration), where the precision of DS-generated labels is fairly good but the recall is too low. The underlying reason is that the facts stored on KB cover too few real facts that the corpus contains. This low-recall issue introduces too many negative instances with basic relation-specific patterns and thus confuses the NRE model in capturing correct patterns. This issue also explains results of $R_1$ in Table 2 that the NRE model trained on DS-generated data achieves high precision but low recall, while DIAG-NRE with reinforced positive patterns can obtain significant improvements. For tasks $R_8$ and $R_9$, the low-recall issues are also severe.

- **False Positives**: This problem is mainly caused by the assumption of DS we described in the introduction section. For example, the precision of DS-generated labels for tasks $R_8$ and $R_{10}$ (/peo./per./place_of_birth) is too low. This low-precision issue means that a large portion of DS-generated positive labels does not indicate the target relation. Thus, this issue inevitably causes the NRE model to absorb some irrelevant patterns. This explanation also corresponds to the fact that we have obtained some negative patterns. By reducing false-positive labels through negative patterns, DIAG-NRE can achieve large precision improvements.

For other tasks, DS-generated labels are relatively good, but the noise issue still exists, major or minor, except for a surprising one, $R_3$ (/loc./loc./contains), where labels automatically created by DS are incredibly accurate. We conjecture that the DS assumption of task $R_3$ is consistent with the written language convention: when mentioning two locations with the containing relation in one sentence, people get used to declaring this relation explicitly.

**Example Patterns & Instances**

Table 4 shows some cases for $R_1$, $R_8$ and $R_{10}$ to show the intuition of our framework. For $R_1$, the positive pattern can
remedy the low coverage problem caused by DS. For $R_s$ and $R_o^j$, besides the help of the positive pattern, the negative pattern can correct many false-positive labels caused by DS. These cases illustrate the ability of DIAG-NRE to diagnose and denoise DS-generated labels.

Conclusion and Future Work
In this paper, we propose a deep pattern diagnosis framework for NRE models trained on DS-generated data. Our framework can extract relation-specific patterns that illustrate predictions of the NRE model. Then, we introduce human priors with only a small set of instances to label and obtain refined patterns. Also, letting human reviewers still handle instance labels instead of writing patterns, we avoid using expensive programmers’ time. After fusing weak labels generated by DS and refined patterns, we can retrain a better NRE model. Extensive experiments on many datasets/relations have shown the effect of our framework to reduce labeling noises over the state-of-the-art baselines.

However, DIAG-NRE is only suitable for NRE models. We note that similar DS approaches are also popular in other tasks, such as (Chen et al. 2017). Therefore, generalizing DIAG-NRE to other tasks in the weakly supervised setting can be an exciting direction to explore.

References
[Angeli et al. 2014] Angeli, G.; Tibshirani, J.; Wu, J.; and Manning, C. D. 2014. Combining distant and partial supervision for relation extraction. In EMNLP, 1556–1567.
[Angeli, Johnson Premkumar, and Manning 2015] Angeli, G.; Johnson Premkumar, M. J.; and Manning, C. D. 2015. Leveraging linguistic structure for open domain information extraction. In ACL, 344–354.
[Bach et al. 2017] Bach, S. H.; He, B.; Ratner, A.; and Ré, C. 2017. Learning the structure of generative models without labeled data. In ICML.
[Bahdanau, Cho, and Bengio 2014] Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.
[Banko et al. 2007] Banko, M.; Cafarella, M. J.; Soderland, S.; Broadhead, M.; and Eitzioni, O. 2007. Open information extraction from the web. In IJCAI, 2670–2676.
[Bengio and others 2009] Bengio, Y., et al. 2009. Learning deep architectures for ai. Foundations and trends® in Machine Learning 2(1):1–127.
[Califf and Mooney 1999] Califf, M. E., and Mooney, R. J. 1999. Relational learning of pattern-match rules for information extraction. In Proceedings of the 16th National Conference on Artificial Intelligence, volume 334.
[Carlson et al. 2010] Carlson, A.; Betteridge, J.; Kisiel, B.; Settles, B.; Hruschka Jr, E. R.; and Mitchell, T. M. 2010. Toward an architecture for never-ending language learning. In AAAI, 3.
[Chen et al. 2017] Chen, Y.; Liu, S.; Zhang, X.; Liu, K.; and Zhao, J. 2017. Automatically labeled data generation for large scale event extraction. In ACL, 409–419.
[Craven, Kumlien, and others 1999] Craven, M.; Kumlien, J.; et al. 1999. Constructing biological knowledge bases by extracting information from text sources. In ISMB, volume 1999, 77–86.
[Cui, Wei, and Zhou 2018] Cui, L.; Wei, F.; and Zhou, M. 2018. Neural open information extraction. In ACL, 407–413.
[Del Corro and Gemulla 2013] Del Corro, L., and Gemulla, R. 2013. Clausie: clause-based open information extraction. In WWW, 355–366.
[Chen et al. 2017] Chen, Y. M.; Liu, S.; Zhang, X.; Liu, K.; and Zhou, B. 2015. Classifying relations by ranking with convolutional neural networks. In ACL, 626–634.
[Feng et al. 2018] Feng, J.; Huang, M.; Zhao, L.; Yang, Y.; and Zhu, X. 2018. Reinforcement learning for relation classification from noisy data. In AAAI.
[Hochreiter and Schmidhuber 1997] Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. Neural computation 9(8):1735–1780.
[Hoffmann et al. 2011] Hoffmann, R.; Zhang, C.; Ling, X.; Zettlemoyer, L.; and Weld, D. S. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In ACL, 541–550.
[Jiang et al. 2017] Jiang, M.; Shang, J.; Cassidy, T.; Ren, X.; Kaplan, L. M.; Hanratty, T. P.; and Han, J. 2017. Metapad: Meta pattern discovery from massive text corpora. In KDD, 877–886.
[Kingma and Ba 2014] Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
[Li, Monroe, and Jurafsky 2016] Li, J.; Monroe, W.; and Jurafsky, D. 2016. Understanding neural networks through representation erasure. arXiv preprint arXiv:1612.08220.
[Lin et al. 2016] Lin, Y.; Shen, S.; Liu, Z.; Luo, H.; and Sun, M. 2016. Neural relation extraction with selective attention over instances. In ACL, 2124–2133.
[Liu et al. 2016] Liu, A.; Soderland, S.; Bragg, J.; Lin, C. H.; Ling, X.; and Weld, D. S. 2016. Effective crowd annotation for relation extraction. In HLT-NAACL, 897–906.
[Liu et al. 2017] Liu, L.; Ren, X.; Zhu, Q.; Zhi, S.; Gui, H.; Ji, H.; and Han, J. 2017. Heterogeneous supervision for relation extraction: A representation learning approach. In EMNLP, 46–56.
[Mausam et al. 2012] Mausam; Schmitz, M.; Soderland, S.; Bart, R.; and Etzioni, O. 2012. Open language learning for information extraction. In EMNLP-CoNLL, 523–534.
[Mintz et al. 2009] Mintz, M.; Bills, S.; Snow, R.; and Jurafsky, D. 2009. Distant supervision for relation extraction without labeled data. In ACL, 1003–1011.
[Miwa and Bansal 2016] Miwa, M., and Bansal, M. 2016. End-to-end relation extraction using lstms on sequences and tree structures. In ACL, 1105–1116.
relational patterns with semantic types. In *EMNLP-CONLL*, 1135–1145.

[Pennington, Socher, and Manning 2014] Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *EMNLP*, 1532–1543.

[Pershina et al. 2014] Pershina, M.; Min, B.; Xu, W.; and Grishman, R. 2014. Infusion of labeled data into distant supervision for relation extraction. In *ACL*, 732–738.

[Qin, Xu, and Wang 2018] Qin, P.; Xu, W.; and Wang, W. Y. 2018. Robust distant supervision relation extraction via deep reinforcement learning. In *ACL*.

[Ratner et al. 2016] Ratner, A. J.; De Sa, C. M.; Wu, S.; Selsam, D.; and Ré, C. 2016. Data programming: Creating large training sets, quickly. In *NIPS*, 3567–3575.

[Ratner et al. 2017] Ratner, A.; Bach, S. H.; Ehrenberg, H.; Fries, J.; Wu, S.; and Ré, C. 2017. Snorkel: Rapid training data creation with weak supervision. *VLDB* 11(3):269–282.

[Riedel, Yao, and McCallum 2010] Riedel, S.; Yao, L.; and McCallum, A. 2010. Modeling relations and their mentions without labeled text. In *ECML*, 148–163.

[Rumelhart, Hinton, and Williams 1986] Rumelhart, D. E.; Hinton, G. E.; and Williams, R. J. 1986. Learning representations by back-propagating errors. *Nature* 323(6088):533.

[Stanovsky and Dagan 2016] Stanovsky, G., and Dagan, I. 2016. Creating a large benchmark for open information extraction. In *EMNLP*, 2300–2305.

[Surdeanu et al. 2012] Surdeanu, M.; Tibshirani, J.; Nallapati, R.; and Manning, C. D. 2012. Multi-instance multi-label learning for relation extraction. In *EMNLP*, 455–465.

[Sutton et al. 2000] Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, 1057–1063.

[Takamatsu, Sato, and Nakagawa 2012] Takamatsu, S.; Sato, I.; and Nakagawa, H. 2012. Reducing wrong labels in distant supervision for relation extraction. In *ACL*, 721–729.

[Varma et al. 2016] Varma, P.; He, B.; Iter, D.; Xu, P.; Yu, R.; De Sa, C.; and Ré, C. 2016. Socratic learning: Augmenting generative models to incorporate latent subsets in training data. *arXiv preprint arXiv:1610.08123*.

[Wang et al. 2016] Wang, L.; Cao, Z.; de Melo, G.; and Liu, Z. 2016. Relation classification via multi-level attention cnns. In *ACL*, 1298–1307.

[Williams 1992] Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.

[Xu et al. 2015] Xu, Y.; Mou, L.; Li, G.; Chen, Y.; Peng, H.; and Jin, Z. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *EMNLP*, 1785–1794.

[Zelenko, Aone, and Richardella 2003] Zelenko, D.; Aone, C.; and Richardella, A. 2003. Kernel methods for relation extraction. *Journal of machine learning research* 3(Feb):1083–1106.