DIAG-NRE: A Neural Pattern Diagnosis Framework for Distantly Supervised Neural Relation Extraction

Shun Zheng\textsuperscript{1}  Xu Han\textsuperscript{2}  Yankai Lin\textsuperscript{2}  Peilin Yu\textsuperscript{3}  Lu Chen\textsuperscript{1}

\textsuperscript{1}Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China
\textsuperscript{2}Department of Computer Science and Technology, Tsinghua University, Beijing, China
\textsuperscript{3}Department of Computer Sciences, University of Wisconsin-Madison, Madison, USA
\textsuperscript{4}AHI Fintech Inc., Beijing, China

\{zhengsl4,hanxu17,linyk14,lchen17\}@mails.tsinghua.edu.cn; peilin@cs.wisc.edu; linghuang@fintec.ai; \{liuzy,weixu\}@tsinghua.edu.cn

Abstract

Pattern-based labeling methods have achieved promising results in alleviating the inevitable labeling noises of distantly supervised neural relation extraction. However, these methods require significant expert labor to write relation-specific patterns, which makes them too sophisticated to generalize quickly. To ease the labor-intensive workload of pattern writing and enable the quick generalization to new relation types, we propose a neural pattern diagnosis framework, DIAG-NRE, that can automatically summarize and refine high-quality relational patterns from noise data with human experts in the loop. To demonstrate the effectiveness of DIAG-NRE, we apply it to two real-world datasets and present both significant and interpretable improvements over state-of-the-art methods. Source codes and data can be found at https://github.com/thunlp/DIAG-NRE.

1 Introduction

Relation extraction aims to extract relational facts from the plain text and can benefit downstream knowledge-driven applications. A relational fact is defined as a relation between a head entity and a tail entity, e.g., \textit{(Letizia\_Moratti, Birthplace, Milan)}. The conventional methods often regard relation extraction as a supervised classification task that predicts the relation type between two detected entities mentioned in a sentence, including both statistical models (Zelenko et al., 2003; Zhou et al., 2005) and neural models (Zeng et al., 2014; dos Santos et al., 2015).

These supervised models require a large number of human-annotated data to train, which are both expensive and time-consuming to collect. Therefore, Craven et al. (1999); Mintz et al. (2009) proposed distant supervision (DS) to automatically generate large-scale training data for relation extraction, by aligning relational facts from a knowledge base (KB) to plain text and assuming that every sentence mentioning two entities can describe their relationships in the KB. As DS can acquire large-scale data without human annotation, it has been widely adopted by recent neural relation extraction (NRE) models (Zeng et al., 2015; Lin et al., 2016).

Although DS is both simple and effective in many cases, it inevitably introduces intolerable labeling noises. As Figure 1 shows, there are two types of error labels, false negatives (FN) and false positives (FP), caused by DS.

![Figure 1: Two types of error labels, false negatives (FN) and false positives (FP), caused by DS.](https://github.com/thunlp/DIAG-NRE)
significantly larger number of noise labels.

To enlarge the impact of human efforts, Ratner et al. (2016); Liu et al. (2017a) proposed to incorporate pattern-based labeling, where the key idea was to regard both DS and pattern-based heuristics as the weak supervision sources and develop a weak-label-fusion (WLF) model to produce denoised labels. However, the major limitation of the WLF paradigm lies in the requirement of human experts to write relation-specific patterns. Unfortunately, writing good patterns is both a high-skill and labor-intensive task that requires experts to learn detailed pattern-composing instructions, examine adequate examples, tune patterns for different corner cases, etc. For example, the *spouse* relation example of Ratner et al. (2016) uses 11 functions with over 20 relation-specific keywords\(^1\). Even worse, when generalizing to a new relation type, we need to repeat the hard manual operations mentioned above again.

To ease the pattern-writing work of human experts and enable the quick generalization to new relation types, we propose a neural pattern diagnosis framework, DIAG-NRE, which establishes a bridge between DS and WLF, for common NRE models. The general workflow of DIAG-NRE, as Figure 2 shows, contains two key stages: 1) *pattern extraction*, extracting potential patterns from NRE models by employing reinforcement learning (RL), and 2) *pattern refinement*, asking human experts to annotate a small set of actively selected examples. Following these steps, we not only minimize the workload and difficulty of human experts by generating patterns automatically, but also enable the quick generalization by only requiring a small number of human annotations. After the processing of DIAG-NRE, we obtain high-quality patterns that are either supportive or unsupportive of the target relation with high probabilities and can feed them into the WLF stage to get denoised labels and retrain a better model. To demonstrate the effectiveness of DIAG-NRE, we conduct extensive experiments on two real-world datasets, where DIAG-NRE not only achieves significant improvements over state-of-the-art methods but also provides insightful diagnostic results for different noise behaviors via refined patterns.

In summary, DIAG-NRE has the following contributions:

\(^1\)https://github.com/HazyResearch/snorkel/tree/master/tutorials/intro

- easing the pattern-writing work of human experts by generating patterns automatically;
- enabling the quick generalization to new relation types by only requiring a small number of human annotations;
- presenting both significant and interpretable performance improvements as well as intuitive diagnostic analyses.

Particularly, for one relation with severe false negative noises, we improve the F1 score by about 40. To the best of our knowledge, we are the first to explicitly reveal and address this severe noise problem for that dataset.

## 2 Related Work

To reduce labeling noises of DS, earlier work attempted to design specific model architectures that can better tolerate labeling noises, such as the multi-instance learning paradigm (Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012; Zeng et al., 2015; Lin et al., 2016; Wu et al., 2017). These models relax the raw assumption of DS by grouping multiple sentences that mention the same entity pair together as a bag and then 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 bag-level models struggled to do sentence-level predictions.

Later work tried to design a dynamic label-adjustment strategy for training (Liu et al., 2017b; Luo et al., 2017). Especially, the most recent work (Feng et al., 2018; Qin et al., 2018) adopted RL to train an agent that interacts with the NRE model to learn how to remove or alter noise labels. These methods work without human intervention by utilizing the consistency and difference between DS-generated labels and model-predicted ones. However, such methods can neither discover...
noise labels that coincide with the model predictions nor explain the reasons for removed or altered labels. As discussed in the introduction, introducing human efforts is a promising direction to contribute both significant and interpretable improvements, which is also the focus of this paper.

As for the pattern-extraction part, we note that there are some methods with similar insights but different purposes. For example, Zhang et al. (2018) improved the performance of the vanilla LSTM (Hochreiter and Schmidhuber, 1997) by utilizing RL to discover structured representations and Li et al. (2016) interpreted the sentiment importance via the prediction of NRE models. Be- sides, Takamatsu et al. (2012) inferred negative syntactic patterns via the example-pattern-relation co-occurrence and removed the false-positive labels accordingly. In contrast, built upon modern neural models, our method not only reduces negative patterns to alleviate false positives but also reinforces positive patterns to address false negatives at the same time.

3 Methodology

Provided with DS-generated data and NRE models trained on them, DIAG-NRE can generate high-quality patterns for the WLF stage to produce denoised labels. As Figure 2 shows, DIAG-NRE contains two key stages in general: pattern extrac- tion (Section 3.2) and pattern refinement (Section 3.3). Moreover, we briefly introduce the WLF paradigm in Section 3.4 for completeness. Next, we start with reviewing the common input-output schema of modern NRE models.

Figure 3: The RL-based pattern-extraction workflow and a typical pattern-induction example, where we induce a pattern for the Birthplace relation via a series of actions (0: retaining, 1: erasing).

3.1 NRE Models

Given an instance \( s \) with \( T \) tokens\(^2\), a common input representation of NRE models is \( x = [x_1, x_2, \ldots, x_T] \), where \( x_i \in \mathbb{R}^{d} \) denotes the embedding of token \( i \) and \( d_t \) is the token embedding size. Particularly, \( x_i \) is the concatenation of the word embedding, \( w_i \in \mathbb{R}^{d_w} \), and position embedding, \( p_i \in \mathbb{R}^{d_p} \), as \( [w_i; p_i] \), to be aware of both semantics and entity positions, where \( d^t = d^w + d^p \). Given the relation type \( r \), NRE models perform different types of tensor manipulations on \( x \) and obtain the predicting probability of \( r \) given the instance \( s \) as \( P_{\phi}(r|x) \), where \( \phi \) denotes model parameters except for the input embedding tables.

3.2 Pattern Extraction

In this stage, we build a pattern-extraction agent to distill relation-specific patterns from NRE models with the aforementioned input-output schema. The basic idea is to erase irrelevant tokens and preserve the raw target prediction simultaneously, which can be modeled as a token-erasing decision process and optimized by RL. Figure 3 shows this RL-based workflow in a general way together with an intuitive pattern-induction example. Next, we elaborate details of this workflow.

Action. The agent takes an action \( a_i \), retaining (0) or erasing (1), for each token of the instance \( s \) and transforms the input representation from \( x \) into \( \hat{x} \). During this process, the column \( i \) of \( x \).

\(^2\)In this paper, we refer to a sentence together with an entity pair as an instance and omit the instance index for brevity.
We define the reward as follows:

\[ R(a|x) = \log \left( \frac{P_{\phi}(r|x)}{P_{\phi}(r|x)} \right) + \eta \cdot (1 - \frac{\hat{T}}{T}), \]

where the total reward is composed of two parts: one is the log-likelihood term to pursue the 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 \( \eta \).

**State.** To be general, the state provided to the agent should be independent of NRE architectures. Moreover, the state needs to incorporate complete information of the current instance. Therefore, in our design, the agent directly employs the input representation \( x \) as the state.

**Agent.** We employ policy-based RL to train a neural-network-based agent that can predict a sequence of actions for an instance to maximize the reward. Our agent network directly estimates \( \pi_{\Theta}(a|x) = \prod_{i=1}^{n} \pi_{\Theta}(a_i|x) \) in a non-autoregressive manner by calculating \( \pi_{\Theta}(a_i|x) \) in parallel, where \( \Theta \) denotes the parameters of the agent network. To enrich the contextual information when deciding the action for each token, we employ the forward and backward LSTM networks to encode \( x \) into \( h \) as

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

\[
\overleftarrow{h} = [\overleftarrow{h}_1, \overleftarrow{h}_2, \cdots, \overleftarrow{h}_T] = \text{Backward-LSTM}(x),
\]

\[
h = [h_1, h_2, \cdots, h_T] = \text{Concatenate}(\overrightarrow{h}, \overleftarrow{h}),
\]

where \( \overrightarrow{h}_i \in \mathbb{R}^{d_h}, \overleftarrow{h}_i \in \mathbb{R}^{d_h}, h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i] \in \mathbb{R}^{2 \times d_h} \), and \( d_h \) denotes the size of LSTM’s hidden state. Then, we employ an attention-based strategy (Bahdanau et al., 2015) to aggregate the contextual information as \( c = [c_1, c_2, \cdots, c_T] \). 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 \( c_j^i / \sum_{k=1}^{T} c_k^i \). Here \( c_j^i \) is computed by a small network as \( c_j^i = v_j^i \tanh(W_j x_i + W_h h_j) \), where \( W_x \in \mathbb{R}^{2d_h \times d_x}, W_h \in \mathbb{R}^{2d_h \times 2d_h} \) and \( v_j \in \mathbb{R}^{2d_h} \) are network parameters. Next, we compute the final representation to infer actions as \( z = [z_1, z_2, \cdots, z_T] \), where for each token \( i \), \( z_i = [x_i; c_i] \in \mathbb{R}^{d_x + 2d_h} \) incorporates semantic, positional and contextual 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 \( o_i = \text{sigmoid}(W_o^T z_i + b_o), W_o \in \mathbb{R}^{d_x + 2d_h} \) and \( b_o \in \mathbb{R}^1 \) are network parameters.

**Optimization.** We employ the REINFORCE algorithm (Williams, 1992) and policy gradient methods (Sutton et al., 2000) to optimize parameters of the agent network, where the key step is to rewrite the gradient formulation and then apply the back-propagation algorithm (Rumelhart et al., 1986) to update network parameters. Specifically, we define our objective as:

\[
L(\Theta) = \mathbb{E}_s \left[ \mathbb{E}_{\pi_{\Theta}(a|x)} R(a|x) \right],
\]

where \( x \) denotes the input representation of the instance \( s \). By taking the derivative of \( J(\Theta) \) with respect to \( \Theta \), we can obtain the gradient \( \nabla_{\Theta} L(\Theta) \) as \( \mathbb{E}_s[\mathbb{E}_{\pi_{\Theta}(a|x)} R(a|x) \nabla_{\Theta} \log \pi_{\Theta}(a|x)] \). Besides, we utilize the \( \epsilon \)-greedy trick to balance exploration and exploitation.

**Pattern Induction.** Given instances and corresponding agent actions, we take the following steps to induce compact patterns. First, to be general, we substitute raw entity pairs with corresponding entity types. Then, we evaluate the agent to obtain retained tokens with the relative distance preserved. To enable the generalized position indication, we divide the relative distance between two adjacent retained tokens into four categories: zero (no tokens between them), short (1-3 tokens),
medium (4-9 tokens) and long (10 or more tokens) distance. For instance, Figure 3 shows a typical pattern-induction example. Patterns with such formats can incorporate multiple kinds of crucial information, such as entity types, key tokens and the relative distance among them.

### 3.3 Pattern Refinement

The above pattern-extraction stage operates at the instance level by producing a pattern for each evaluated instance. However, after aggregating available patterns at the dataset level, there inevitably exist redundant ones. Therefore, we design a pattern hierarchy to merge redundant patterns. Afterward, we can introduce human experts into the workflow by asking them to annotate a small number of actively selected examples. Figure 4 shows the general workflow of this stage.

**Pattern Hierarchy.** To identify redundant patterns, we group multiple instances with the same pattern and build a pattern hierarchy by the matching statistics. In this hierarchy, the parent pattern should cover all instances matched by the child pattern. As the parent pattern already has sufficient relation-supporting signals, we can omit child patterns for human annotation. Moreover, the number of instances from which the pattern can be induced is closely related to the pattern representativeness. Therefore, we follow the decreasing order of this number to select top \( n_r \) most representative patterns for human annotation.

**Human Annotation.** To quantitatively evaluate the pattern quality, we adopt an approximate method by randomly selecting \( n_a \) pattern-matched instances and annotating them manually. Thus, for each relation type, we end up with \( n_r \times n_a \) human-annotated instances. We assign patterns with the accuracy higher than \( p_l \) and lower than \( p_h \) into the positive pattern set and the negative pattern set, respectively, to serve the WLF stage. In practice, users can tune these hyper-parameters (\( n_r, n_a, p_h \) and \( p_l \)) accordingly for different applications, such as increasing \( p_h \) to prefer precision. While in this paper, to show the wide applicability and robustness of DIAG-NRE, we demonstrate that a single configuration can handle all 14 relation types.

### 3.4 Weak Label Fusion

The WLF model aims to fuse weak labels from multiple labeling sources, including both DS and patterns, to produce denoised labels. In this paper, we adopt *data programming* (DP) (Ratner et al., 2016) at our WLF model. The input unit of DP is called *labeling function* (LF), which takes one instance and emits a label (+1: *positive*, -1: *negative* or 0: *unknown*). 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 estimate parameters of DP on the small set of human-annotated labels with a closed-form solution (see the appendix for detailed formulations). With the help of DP, we get denoised labels to retrain a better model. Note that designing better generic WLF models is still a hot research topic (Varma et al., 2016; Bach et al., 2017; Liu et al., 2017a) but outside the scope of this work, which is automatically generating patterns to ease human’s work.

### 4 Experiments

In this section, we present experimental results and comprehensive analyses to demonstrate the effectiveness of DIAG-NRE.

#### 4.1 Experimental Setup

**Evaluation.** To clearly show the different noise behaviours for various relation types, we treat each relation prediction task as a single binary classification problem, that is predicting the existing or not of that relation for a given instance. Different from previous studies, we report relation-specific metrics (Precision, Recall and F1 scores, all in the percentage format) and macro-averaged ones at the dataset level, because the distribution of relation types is extremely imbalanced and the micro-averaged evaluation inevitably overlooks noisy-labeling issues of many relation types. Moreover, we only utilize human-annotated test data to evaluate models trained on noise labels, as Ratner et al. (2016); Liu et al. (2016) did. The reason is that the
We implement DIAG-NRE

with five random seeds, ranging from 0 to 4, for parameters exactly same, we run the NRE model used to train multiple agents that tend to squeeze patterns with different granularities and combine outputs of all agents to serve the pattern-refinement stage. To speed up the agent training, we use filtered instances by taking the top 10,000 ones with the highest prediction probabilities. At the pattern-refinement stage, hyper-parameters include $n_r = 20$, $n_a = 10$, $p_b = 0.8$ and $p_l = 0.1$. Thus, for each task, we get 200 human-annotated instances (about 0.05% of the entire train set) and at most 20 patterns for the WLF stage.

4.2 Performance Comparisons

Based on the above hyper-parameters, DIAG-NRE together with the WLF model can produce de-noised labels to retrain a better NRE model. Next, we present the overall performance comparisons of NRE models trained with different labels.

Baselines. We adopt the following baselines: 1) Distant Supervision, the vanilla DS described in Mintz et al. (2009), 2) Gold Label Mix (Liu et al., 2016), mixing human-annotated high-quality labels with DS-generated noise labels, and 3) RLRE (Feng et al., 2018), building an instance-selection agent to select correct-labeled ones by only interacting with NRE models trained on noise labels. Specifically, for Gold Label Mix, we use the same 200 labels obtained at the pattern-refinement stage as the high-quality labels. To focus on the impact of training labels produced with different methods, besides for fixing all hyper-parameters exactly same, we run the NRE model with five random seeds, ranging from 0 to 4, for each case and present the averaged scores.

Overall Results. Table 2 shows the overall results with precision (P), recall (R) and F1 scores. For a majority of tasks suffering large labeling noises, including $R_1$, $R_4$, $R_5$, $R_8$, $R_9$ and $R_2^*$, we improve the F1 score by 5.0 over the best baseline. Notably, the F1 improvement for task $R_1$ has

<table>
<thead>
<tr>
<th>TID</th>
<th>Relation Abbreviation</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_0$</td>
<td>Bus./Company</td>
<td>5.3k</td>
<td>186</td>
</tr>
<tr>
<td>$R_1$</td>
<td>Loc./Admin.Div.</td>
<td>4.9k</td>
<td>180</td>
</tr>
<tr>
<td>$R_2$</td>
<td>Loc./Capital</td>
<td>5.3k</td>
<td>20</td>
</tr>
<tr>
<td>$R_3$</td>
<td>Loc./Contains</td>
<td>44.6k</td>
<td>263</td>
</tr>
<tr>
<td>$R_4$</td>
<td>Loc./Country</td>
<td>4.9k</td>
<td>89</td>
</tr>
<tr>
<td>$R_5$</td>
<td>Loc./Neighbor</td>
<td>5.6k</td>
<td>55</td>
</tr>
<tr>
<td>$R_6$</td>
<td>Peo./National</td>
<td>7.5k</td>
<td>84</td>
</tr>
<tr>
<td>$R_7$</td>
<td>Peo./PlaceLived</td>
<td>6.7k</td>
<td>230</td>
</tr>
<tr>
<td>$R_8$</td>
<td>Peo./Birthplace</td>
<td>3.1k</td>
<td>16</td>
</tr>
<tr>
<td>$R_9$</td>
<td>Peo./Deathplace</td>
<td>1.9k</td>
<td>19</td>
</tr>
<tr>
<td>$R_2^*$</td>
<td>Peo./National</td>
<td>107k</td>
<td>1.8k</td>
</tr>
<tr>
<td>$R_3^*$</td>
<td>Peo./PlaceLived</td>
<td>20.9k</td>
<td>3.8k</td>
</tr>
<tr>
<td>$R_4^*$</td>
<td>Peo./Birthplace</td>
<td>15.3k</td>
<td>458</td>
</tr>
<tr>
<td>$R_5^*$</td>
<td>Peo./Deathplace</td>
<td>5.7k</td>
<td>1.3k</td>
</tr>
</tbody>
</table>

Table 1: The total 14 relation prediction tasks with corresponding task IDs (TIDs), relation abbreviations and the number of positive labels in the train and test sets.
### 4.3 Pattern-based Diagnostic Results

Besides for improving the extraction performance, DIAG-NRE can interpret different noise effects caused by DS via refined patterns, as Table 3 shows. Next, we elaborate these diagnostic results and the corresponding performance degradation of NRE models from two perspectives: false negatives (FN) and false positives (FP).

**FN.** A typical example of FN is task $R_1$ (Administrative Division), where the precision of DS-generated labels is fairly good but the recall is too low. The underlying reason is that the relational facts stored in the KB cover too few real facts actually contained by the corpus. This low-recall issue introduces too many negative instances with common relation-supporting patterns and thus confuses the NRE model in capturing correct features. This issue also explains the specific reason. We conjecture that this behavior is due to the gap between maximizing the likelihood of the NRE model and the ground-truth instance selection. In contrast, DIAG-NRE can contribute both stable and interpretable improvements with the help of human-readable patterns.

### Table 3: Total diagnostic results, where columns contain the precision, recall and accuracy of DS-generated labels evaluated on 200 human-annotated labels as well as the number of positive and negative patterns preserved after the pattern-refinement stage, and we underline some cases in which DS performs poorly.

<table>
<thead>
<tr>
<th>TID</th>
<th>Prec.</th>
<th>Recall</th>
<th>Acc.</th>
<th>#Pos.</th>
<th>#Neg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_0$</td>
<td>100.0</td>
<td>81.8</td>
<td>82.0</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>$R_1$</td>
<td>93.9</td>
<td>33.5</td>
<td>36.2</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>$R_2$</td>
<td>75.7</td>
<td>88.0</td>
<td>76.5</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>$R_3$</td>
<td>100.0</td>
<td>91.4</td>
<td>92.0</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>$R_4$</td>
<td>93.3</td>
<td>72.4</td>
<td>80.9</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>$R_5$</td>
<td>93.8</td>
<td>77.3</td>
<td>86.5</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>$R_6$</td>
<td>88.3</td>
<td>76.9</td>
<td>75.1</td>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>$R_7$</td>
<td>91.9</td>
<td>64.6</td>
<td>64.0</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>$R_8$</td>
<td>29.3</td>
<td>30.4</td>
<td>60.0</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>$R_9$</td>
<td>66.7</td>
<td>38.1</td>
<td>74.4</td>
<td>6</td>
<td>11</td>
</tr>
<tr>
<td>$R_{10}$</td>
<td>81.8</td>
<td>90.7</td>
<td>81.0</td>
<td>7</td>
<td>0</td>
</tr>
<tr>
<td>$R_{11}$</td>
<td>93.5</td>
<td>70.7</td>
<td>68.3</td>
<td>17</td>
<td>1</td>
</tr>
<tr>
<td>$R_{12}$</td>
<td>35.0</td>
<td>70.0</td>
<td>60.0</td>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>$R_{13}$</td>
<td>87.5</td>
<td>95.4</td>
<td>67.7</td>
<td>12</td>
<td>5</td>
</tr>
</tbody>
</table>

Reaching 40. For some tasks with fewer noises, including $R_0$, $R_7$, $R_1$ and $R_{12}$, our method can obtain small improvements. For a few tasks, such as $R_3$, $R_6$ and $R_{12}$, only using DS is sufficient to train competitive models. In such cases, fusing other weak labels may have negative effects, but these side effects are small. The detailed reasons for these improvements will be elaborated together with the diagnostic results in Section 4.3. Another interesting observation is that RLRE performs the best result on tasks $R_2$ and $R_{12}$ 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 difficult to be interpreted, 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 ground-truth instance selection. In contrast, DIAG-NRE can contribute both stable and interpretable improvements with the help of human-readable patterns.
Table 4: Positive (Pos.), negative (Neg.) patterns and associated examples with labels produced by different methods. For each pattern, we present “DS Label” as the number of DS-generated positive labels over the number of pattern-matched instances. For RLRE, None means the instance is removed. For DIAG-NRE, we present the soft label produced by the WLF model.

4.4 Incremental Diagnosis

In addition to the performance comparisons based on 200 human-annotated instances, we show the incremental diagnosis ability of DIAG-NRE by gradually increasing the number of human annotations from 10 to 200. As Figure 5 shows, where we pick those tasks (three from NYT and two from UW) suffering large labeling noises, most tasks experience a rapid improvement phase with the help of high-quality patterns automatically generated by DIAG-NRE and then enter a saturate phase where adding annotations does not contribute much. This saturation accords with the intuition that high-quality relational patterns are often limited. The only exception is task $R_9$ that drops first and then increases again, the reason is that the fully automatic pattern refinement of DIAG-NRE produces one incorrect pattern accidentally, while later patterns alleviate this mistake. Actually, in practice, users can further curate patterns generated by DIAG-NRE to get even better results, which can also be much easier and quicker than writing patterns from scratch.

4.5 Case Studies

Table 4 shows five pattern examples from three tasks. For task $R_1$, the positive pattern can remedy the extremely low coverage caused by DS. For tasks $R_8$ and $R_9$, besides for the help of the positive pattern, the negative pattern can correct many
FP labels caused by DS. These cases intuitively illustrate the ability of DIAG-NRE to diagnose and denoise DS-generated labels.

5 Conclusion and Future Work

In this paper, we propose a neural pattern diagnosis framework, DIAG-NRE, to diagnose and improve NRE models trained on DS-generated data. DIAG-NRE not only eases the hard pattern-writing work of human experts by generating patterns automatically, but also enables the quick generalization to new relation types by only requiring a small number of human annotations. Coupled with the WLF model, DIAG-NRE can produce denoised labels to retrain a better NRE model. Extensive experiments with comprehensive analyses demonstrate that DIAG-NRE can contribute both significant and interpretable improvements.

For the future work, we plan to extend DIAG-NRE to other DS-based applications, such as question answering (Lin et al., 2018), event extraction (Chen et al., 2017), etc.

Acknowledgements

This work is supported in part by the National Natural Science Foundation of China (NSFC) Grant 61532001, 61572273, 61532010, Tsinghua Initiative Research Program Grant 20151080475, and gift funds from Ant Financial and Nanjing Turing AI Institute. Xu Han is also supported by 2018 Tencent Rhino-Bird Elite Training Program.

References

Gabor Angeli, Julie Tibshirani, Jean Wu, and Christopher D Manning. 2014. Combining distant and partial supervision for relation extraction. In EMNLP.


Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In ICLR.

Mary Elaine Califf and Raymond J. Mooney. 1999. Relational learning of pattern-match rules for information extraction. In AAAI.


Yubo Chen, Shulin Liu, Xiang Zhang, Kang Liu, and Jun Zhao. 2017. Automatically labeled data generation for large scale event extraction. In ACL.

Mark Craven, Johan Kumlien, et al. 1999. Constructing biological knowledge bases by extracting information from text sources. In ISMB.

Jun Feng, Minlie Huang, Li Zhao, Yang Yang, and Xiaoyan Zhu. 2018. Reinforcement learning for relation classification from noisy data. In AAAI.


Meng Jiang, Jingbo Shang, Taylor Cassidy, Xiang Ren, Lance M Kaplan, Timothy P Hanratty, and Jiawei Han. 2017. Metapad: meta pattern discovery from massive text corpora. In KDD.


Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In ACL.

Angli Liu, Stephen Soderland, Jonathan Bragg, Christopher H Lin, Xiao Ling, and Daniel S Weld. 2016. Effective crowd annotation for relation extraction. In NAACL-HLT.

Liyuan Liu, Xiang Ren, Qi Zhu, Shi Zhi, Huan Gui, Heng Ji, and Jiawei Han. 2017a. Heterogeneous supervision for relation extraction: a representation learning approach. In EMNLP.


Bingfeng Luo, Yansong Feng, Zheng Wang, Zhanxing Zhu, Songfang Huang, Rui Yan, and Dongyan Zhao. 2017. Learning with noise: enhance distantly supervised relation extraction with dynamic transition matrix. In ACL.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In ACL.


Maria Pershina, Bonan Min, Wei Xu, and Ralph Grishman. 2014. Infusion of labeled data into distant supervision for relation extraction. In *ACL*.


Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *ECML*.


Ce Zhang, Feng Niu, Christopher Ré, and Jude Shavlik. 2012. Big data versus the crowd: looking for relationships in all the right places. In *ACL*.


A Appendices

In the appendices, we introduce formulation details of the weak-label-fusion (WLF) model and the hyper-parameters for our neural relation extraction (NRE) model.

A.1 Weak Label Fusion

As mentioned in the main body, we employ the data programming (DP) (Ratner et al., 2016) as our WLF model. DP proposed an abstraction of the weak label generator, named as the labeling function (LF), which can incorporate both DS and pattern-based heuristics. Typically, for a binary classification task, an LF is supposed to produce +1 or -1, LFs of positive patterns generate +1 or 0, and LFs of negative patterns generate -1 or 0.

Given a labeling function, we can write the joint probability of weak labels \( L^s \) and the true label \( Y^s \) for instance \( s \), \( P_{\alpha, \beta}(L^s, Y^s) \), as

\[
\frac{1}{2} \prod_{i=1}^{m} (\beta_i \alpha_i 1_{L^s_i = Y^s} + \beta_i (1 - \alpha_i) 1_{L^s_i = -Y^s}) + (1 - \beta_i) 1_{(L^s_i = 0)},
\]

where each \( L^s_i \in \{-1, 0, +1\} \) denotes the weak label generated for instance \( s \) by the \( i^{th} \) labeling function, and \( \alpha \) and \( \beta \) are model parameters to be estimated.

Originally, Ratner et al. (2016) conducted the unsupervised parameter estimation based on unlabeled data by solving

\[
\max_{\alpha, \beta} \sum_{s \in S} \log \left( \sum_{Y^s} P_{\alpha, \beta}(L^s, Y^s) \right).
\]

Different from the general DP that treats each LF with the equal prior, we have strong priors that patterns produced by DIAG-NRE are either supportive or unsupportive of the target relation with high probabilities. Therefore, in our case, we directly employ the small labeled set \( S_L \) obtained at the pattern-refinement stage 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

\[
\begin{align*}
\alpha_i &= \frac{\sum_{s \in S_L} 1_{L^s_i = Y^s}}{\sum_{s \in S_L} 1_{(L^s_i = Y^s)} + 1_{(L^s_i = -Y^s)}}, \\
\beta_i &= \frac{\sum_{s \in S_L} 1_{(L^s_i = Y^s)} + 1_{(L^s_i = -Y^s)}}{|S_L|},
\end{align*}
\]

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 denoised soft label to train a better NRE model, just as Ratner et al. (2016) did.

A.2 Hyper-parameters of the NRE model

For the NRE model, we implement a simple yet effective LSTM-based architecture described in (Zhou et al., 2016). We conduct the hyper-parameter search via cross-validation and adopt the following configurations that can produce pretty good results for all 14 tasks. First, the word embedding table (\( d^w = 100 \)) is initialized with Glove vectors (Pennington et al., 2014), the size of the position vector (\( d^p \)) is 5, the maximum length of the encoded relative distance is 60, and we follow (Zeng et al., 2015; Lin et al., 2016) to randomly initialize these position vectors. Besides, the LSTM hidden size is 200, and the dropout probabilities at the embedding layer, the LSTM layer and the last layer are 0.3, 0.3 and 0.5, respectively. During training, we employ the Adam (Kingma and Ba, 2014) optimizer with the learning rate of 0.001 and the batch size of 50. Moreover, we select the best epoch according to the score on the validation set.

Notably, we observe that when training on data with large labeling noises, different parameter initializations can heavily influence the extraction performance of trained models. Therefore, as mentioned in the main body, to clearly and fairly show the actual impact of different types of training labels, we restart the training of NRE models with 5 random seeds, ranging from 0 to 4, for each case and report the averaged scores.