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

Shun Zheng[1] **Xu Han**[2] **Yankai Lin**[2] **Peilin Yu**[3] **Lu Chen**[1]
Ling Huang[1,4] **Zhiyuan Liu**[2] **Wei Xu**[1]

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

{zhengs14,hanxu17,linyk14,lchen17}@mails.tsinghua.edu.cn; peilin@cs.wisc.edu;
linghuang@fintec.ai; {liuzy,weixu}@tsinghua.edu.cn;

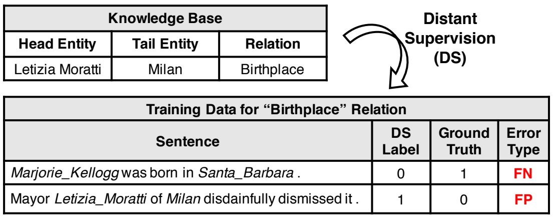
Abstract

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., (*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).



Knowledge Base		
Head Entity	Tail Entity	Relation
Letizia Moratti	Milan	Birthplace

Training Data for “Birthplace” Relation			
Sentence	DS Label	Ground Truth	Error Type
Marjorie_Kellogg was born in *Santa_Barbara* .	0	1	FN
Mayor *Letizia_Moratti* of *Milan* disdainfully dismissed it .	1	0	FP

Figure 1: Two types of error labels, false negatives (FN) and false positives (FP), caused by DS.

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 and false positives. The reason for false negatives is that a sentence does describe two entities about a target relation, but the fact has not been covered by the KB yet. While for false positives, it is because not all sentences mentioning entity pairs actually express their relations in the KB. The noisy-labeling problem can become severe when the KB and text do not match well and as a result heavily weaken the model performance (Riedel et al., 2010).

Recent research has realized that introducing appropriate human efforts is essential for reducing such labeling noises. For example, 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 noise labels. However, they found that only sufficiently large and high-quality human labels can bring notable improvements, because there are

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1419–1429
Florence, Italy, July 28 - August 2, 2019.

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:

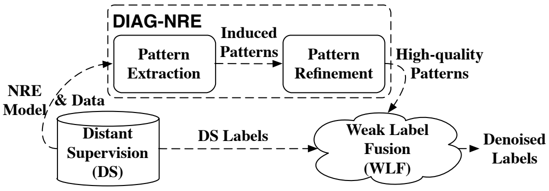


Figure 2: An overview of DIAG-NRE.

- 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 0.4. 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

[1] https://github.com/HazyResearch/snorkel/tree/master/tutorials/intro

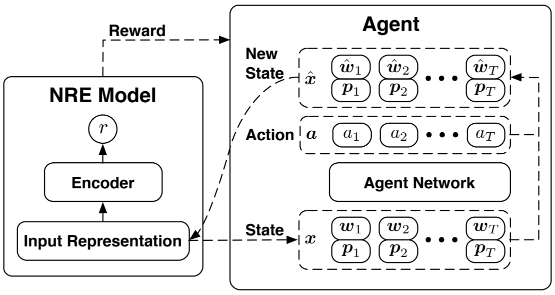
noise labels that coincide with the model predictions nor explain the reasons for removed or altered labels. As discussed in the introduction, introducing human efforts is a promising direction to contribute both significant and interpretable improvements, which is also the focus of this paper.

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 prediction of neural models by employing RL to find the decision-changing phrases. However, NRE models are unique because we only care about the semantic inter-entity relation mentioned in the sentence. To the best of our knowledge, we are the first to extract patterns from NRE models by RL.

We also note that the relational-pattern mining has been extensively studied (Califf and Mooney, 1999; Carlson et al., 2010; Nakashole et al., 2012; Jiang et al., 2017). Different from those studies, our pattern-extraction method 1) is simply based on RL, 2) does not rely on any lexical or syntactic annotation, and 3) can be aware of the pattern importance via the prediction of NRE models. Besides, 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 extraction (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.



Pattern-induction Example						
Entities	PER				CITY	
Tokens	*Joachim_Fest*	was	born	in	*Berlin*	.
Actions	0	1	0	0	*0*	1
Pattern	ENTITY1:PER PAD{1,3} born in ENTITY2:CITY					

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 $\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T]$, where $\boldsymbol{x}_i \in \mathbb{R}^{d^x}$ denotes the embedding of token i and d^x is the token embedding size. Particularly, $\boldsymbol{x}_i$ is the concatenation of the word embedding, $\boldsymbol{w}_i \in \mathbb{R}^{d^x}$, and position embedding, $\boldsymbol{p}_i \in \mathbb{R}^{d^p}$, as $[\boldsymbol{w}_i; \boldsymbol{p}_i]$, to be aware of both semantics and entity positions, where $d^x = d^w + d^p$. Given the relation type r, NRE models perform different types of tensor manipulations on $\boldsymbol{x}$ and obtain the predicting probability of r given the instance s as $P_\phi(r|\boldsymbol{x})$, where ϕ 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 $\boldsymbol{x}$ into $\hat{\boldsymbol{x}}$. During this process, the column i of $\boldsymbol{x}$,

[2] In this paper, we refer to a sentence together with an entity pair as an instance and omit the instance index for brevity.

$\boldsymbol{x}_i = [\boldsymbol{w}_i; \boldsymbol{p}_i]$, corresponding to the token i of raw instance s, is transformed into $\hat{\boldsymbol{x}}_i = [\hat{\boldsymbol{w}}_i; \boldsymbol{p}_i]$, where the position vectors are left untouched and the new word vector $\hat{\boldsymbol{w}}_i$ is adjusted based on the action taken by the agent. For the *retaining* action, we retain the raw word vector as $\hat{\boldsymbol{w}}_i = \boldsymbol{w}_i$. While for *erasing*, we set $\hat{\boldsymbol{w}}_i$ to be all zeros to remove the semantic meaning. After taking a sequence of actions, $\boldsymbol{a} = [a_1; a_2; \cdots; a_T]$, we get the transformed representation $\hat{\boldsymbol{x}}$ with $\hat{T}$ tokens retained.

Reward. Our purpose is to find the most simplified sequence that preserves the raw prediction confidence. Therefore, given the raw input representation x and the corresponding action vector $\boldsymbol{a}$, we define the reward as follows:

$$R(\boldsymbol{a}|\boldsymbol{x}) = \underbrace{\log\left(\frac{P_\phi(r|\hat{\boldsymbol{x}})}{P_\phi(r|\boldsymbol{x})}\right)}_{\text{Prediction Confidence}} + \eta \cdot \underbrace{(1 - \hat{T}/T)}_{\text{Sparsity}},$$

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 η.

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 $\boldsymbol{x}$ 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 $\pi_\Theta(\boldsymbol{a}|\boldsymbol{x}) = \prod_{i=1}^{T} \pi_\Theta(a_i|\boldsymbol{x})$ in a non-autoregressive manner by calculating $\pi_\Theta(a_i|\boldsymbol{x})$ in parallel, where Θ 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 $\boldsymbol{x}$ into $\boldsymbol{h}$ as

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

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

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

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 $\boldsymbol{c} = [c_1, c_2, \cdots, c_T]$. 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 α_j^i is calculated by $e_j^i / (\sum_{k=1}^{T} e_k^i)$. Here e_j^i is computed by a small network as $e_j^i = v_\alpha^\top \tanh(W_x x_i + W_h h_j)$, where $W_x \in \mathbb{R}^{2d_h \times d_x}$, $W_h \in \mathbb{R}^{2d_h \times 2d_h}$ and $v_\alpha \in \mathbb{R}^{2d_h}$ are network parameters. Next, we compute the final representation to infer actions as $\boldsymbol{z} = [z_1, z_2, \cdots, z_T]$, 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

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

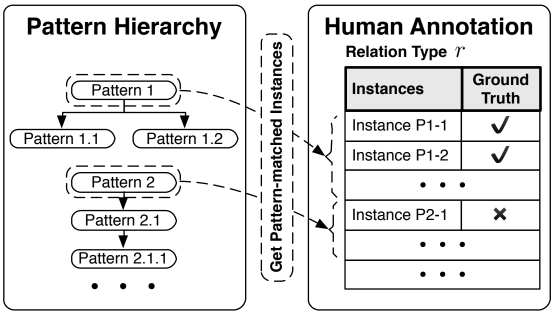
where $o_i = \text{sigmoid}(W_o^\top z_i + b_o)$, $W_o \in \mathbb{R}^{d^x + 2d^h}$ and $b_o \in \mathbb{R}^1$ are network parameters.

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:

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

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



Instances	Ground Truth
Instance P1-1	✓
Instance P1-2	✓
• • •	
Instance P2-1	×
• • •	
• • •	

Figure 4: The human-in-the-loop pattern refinement.

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 $n_r * n_a$ human-annotated 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 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 patters 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

	TID	Relation Abbreviation	Train	Test
NYT	R_0	Bus./Company	5.3k	186
	R_1	Loc./Admin._Div.	4.9k	180
	R_2	Loc./Capital	5.3k	20
	R_3	Loc./Contains	44.6k	263
	R_4	Loc./Country	4.9k	89
	R_5	Loc./Neighbor.	5.6k	55
	R_6	Peo./National.	7.5k	84
	R_7	Peo./Place_Lived	6.7k	230
	R_8	Peo./Birthplace	3.1k	16
	R_9	Peo./Deathplace	1.9k	19
UW	R_6^u	Peo./National.	107k	1.8k
	R_7^u	Peo./Place_Lived	20.9k	3.8k
	R_8^u	Peo./Birthplace	15.3k	458
	R_9^u	Peo./Deathplace	5.7k	1.3k

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. The train set, generated by DS, contains $452,223$ and $395,738$ instances for NYT and UW, respectively. The test set, annotated by the human, contains $1,027$ and $15,622$ instances for NYT and UW, respectively.

severe labeling noises of many relation types heavily weaken the reliability of the DS-based *held-out evaluation* (Mintz et al., 2009), which cannot judge the performance accurately.

Data & Tasks. We select top ten relation types with enough coverage (over $1,000$ instances) from the NYT dataset (Riedel et al., 2010)[3] and all four relation types from the UW dataset (Liu et al., 2016)[4]. Originally, the NYT dataset contains a train set and a test set both generated by DS with $522,611$ and $172,448$ instances, respectively; the UW dataset contains a train set generated by DS, a crowd-annotated set and a minimal human-annotated test set with $676,882$, $18,128$ and 164 instances, respectively. To enable the reliable evaluation based on human annotations, for the NYT dataset, we randomly select up to 100 instances per relation (including the special unknown relation NA) from the test set and manually annotate them; while for the UW dataset, we directly utilize the crowd-annotated set (disjoint from the train set) with the broad coverage and very high quality as the ground truth. Table 1 summaries detailed statistics of these 14 tasks.

Hyper-parameters. We implement DIAG-NRE based on Pytorch[5] and directly utilize its default initialization for neural networks. For the NRE model, we adopt a simple yet effective LSTM-based architecture described in Zhou et al. (2016) and adopt widely-used hyper-parameters (see the appendix for details). As for DIAG-NRE, we use the following configuration for all 14 tasks. For the agent network, 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. At the pattern-extraction stage, we use $\epsilon = 0.1$ and alter η in $\{0.05, 0.1, 0.5, 1.0, 1.5\}$ 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 $n_r = 20$, $n_a = 10$, $p_h = 0.8$ 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 denoised 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_8^u, we improve the F1 score by 5.0 over the best baseline. Notably, the F1 improvement for task R_1 has

[3] http://iesl.cs.umass.edu/riedel/ecml/

[4] https://www.cs.washington.edu/ai/gated_instructions/naacl_data.zip

[5] https://pytorch.org/

TID	Distant Supervision			Gold Label Mix			RLRE			DIAG-NRE				
	P.	**R.**	**F1**	**P.**	**R.**	**F1**	**P.**	**R.**	**F1**	**P.**	**R.**	**F1**	**Inc-DS**	**Inc-Best**
R_0	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**
R_1	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**
R_2	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
R_3	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
R_4	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**
R_5	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**
R_6	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
R_7	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**
R_8	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**
R_9	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**
Avg.	80.5	50.4	55.1	81.2	53.0	57.8	84.1	47.3	54.8	82.5	60.5	**65.6**	**+10.5**	**+6.5**
R_6^u	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
R_7^u	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**
R_8^u	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**
R_9^u	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**
Avg.	52.0	57.4	49.4	53.1	57.3	50.0	52.0	57.3	49.4	55.3	60.2	**53.7**	**+4.3**	**+3.5**

Table 2: Overall results for 14 tasks, where we present relation-specific scores, the macro-averaged ones (Avg.), the F1 improvement of DIAG-NRE over the vanilla DS (Inc-DS) and the best baseline (Inc-Best), and we highlight the best F1 for each task and the significant improvements.

TID	Prec.	Recall	Acc.	#Pos.	#Neg.
R_0	100.0	81.8	82.0	20	0
R_1	93.9	$\underline{33.5}$	$\underline{36.2}$	18	0
R_2	75.7	88.0	76.5	9	5
R_3	100.0	91.4	92.0	20	0
R_4	93.3	72.4	80.9	10	2
R_5	93.8	77.3	86.5	15	0
R_6	88.3	76.9	75.1	14	0
R_7	91.9	64.6	64.0	20	0
R_8	$\underline{29.3}$	$\underline{30.4}$	60.0	4	10
R_9	66.7	$\underline{38.1}$	74.4	6	11
R_6^u	81.8	90.7	81.0	7	0
R_7^u	93.5	70.7	68.3	17	1
R_8^u	$\underline{35.0}$	70.0	60.0	4	15
R_9^u	87.5	59.2	67.7	12	5

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.

reached 40. For some tasks with fewer noises, including R_0, R_7, R_7^u and R_9^u, our method can obtain small improvements. For a few tasks, such as R_3, R_6 and R_6^u, 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 yields the best result on tasks R_2 and R_6^u 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.

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 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 im-

TID	Patterns & Matched Examples	DS	RLRE	DIAG-NRE
R_1	**Pos. Pattern:** `in ENTITY2:CITY PAD{1,3} ENTITY1:COUNTRY` **(DS Label:** 382 / 2072)			
	Example: He will , however , perform this month in *Rotterdam* , the *Netherlands* , and Prague .	0	`None`	0.81
R_8	**Pos. Pattern**: `ENTITY1:PER PAD{1,3} born PAD{1,3} ENTITY2:CITY` **(DS Label:** 44 / 82)			
	Example: *Marjorie_Kellogg* was born in *Santa_Barbara* .	0	0	1.0
	Neg. Pattern: `mayor ENTITY1:PER PAD{1,3} ENTITY2:CITY` **(DS Label:** 21 / 62)			
	Example: Mayor *Letizia_Moratti* of *Milan* disdainfully dismissed it .	1	1	0.0
R_9^u	**Pos. Pattern:** `ENTITY1:PER died PAD{4,9} ENTITY2:CITY` **(DS Label:** 66 / 108)			
	Example: *Dahm* died Thursday at an assisted living center in *Huntsville* ...	0	0	1.0
	Neg. Pattern: `ENTITY1:PER PAD{4,9} rally PAD{1,3} ENTITY2:CITY` **(DS Label:** 40 / 87)			
	Example: *Bhutto* vowed to hold a rally in *Rawalpindi* on Friday ...	1	1	0.0

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.

provements due to much higher recall. For tasks R_8 (`Birthplace`) and R_9 (`Deathplace`), we observe the similar low-recall issues.

FP. The FP errors are mainly caused by the assumption of DS described in the introduction. For example, the precision of DS-generated labels for tasks R_8 and R_8^u is too low. This low precision means that a large portion of DS-generated positive labels do 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 labels with FP errors 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 task R_3 (`Contains`), where labels automatically generated by DS are incredibly accurate. We conjecture the reason for such high-quality labeling is that for task R_3, the DS assumption 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.

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.

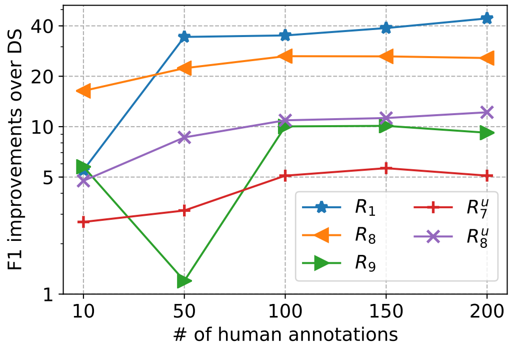


Figure 5: The F1 improvements of DIAG-NRE over DS with the increased number of human annotations.

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^u, 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*.

Stephen H. Bach, Bryan He, Alexander Ratner, and Christopher Ré. 2017. Learning the structure of generative models withut labeled data. In *ICML*.

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*.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. 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*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL*.

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*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*.

Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distantly supervised open-domain question answering. In *ACL*.

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*.

Tianyu Liu, Kexiang Wang, Baobao Chang, and Zhifang Sui. 2017b. A soft-label method for noise-tolerant distantly supervised relation extraction. 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*.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Patty: a taxonomy of relational patterns with semantic types. In *EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: global vectors for word representation. In *EMNLP*.

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

Pengda Qin, Weiran Xu, and William Yang Wang. 2018. Robust distant supervision relation extraction via deep reinforcement learning. In *ACL*.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: creating large training sets, quickly. In *NIPS*.

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

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by backpropagating errors. *Nature*.

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *ACL*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *EMNLP*.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *ACL*.

Paroma Varma, Bryan He, Dan Iter, Peng Xu, Rose Yu, Christopher De Sa, and Christopher Ré. 2016. Socratic learning: augmenting generative models to incorporate latent subsets in training data. *arXiv preprint arXiv:1610.08123*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*.

Yi Wu, David Bamman, and Stuart Russell. 2017. Adversarial training for relation extraction. In *EMNLP*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *JMLR*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *COLING*.

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*.

Tianyang Zhang, Minlie Huang, and Li Zhao. 2018. Learning structured representation for text classification via reinforcement learning. In *AAAI*.

Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *ACL*.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. 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 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.

Given m labeling functions, we can write the joint probability of weak labels $\boldsymbol{L}^s$ and the true label $Y^s \in \{-1, +1\}$ for instance s, $P_{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{L}^s, Y^s)$, as

$$\frac{1}{2}\prod_{i=1}^{m}(\beta_i\alpha_i\mathbf{1}_{\{L_i^s=Y^s\}} + \beta_i(1-\alpha_i)\mathbf{1}_{\{L_i^s=-Y^s\}} \\ + (1-\beta_i)\mathbf{1}_{\{L_i^s=0\}}),$$

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

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

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

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 $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ by solving

$$\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \sum_{s\in S_L} \log P_{\boldsymbol{\alpha},\boldsymbol{\beta}}(\boldsymbol{L}^s, Y^s),$$

where the closed-form solutions are

$$\alpha_i = \frac{\sum_{s\in S_L} \mathbf{1}_{\{L_i^s=Y^s\}}}{\sum_{s\in S_L}\left[\mathbf{1}_{\{L_i^s=Y^s\}} + \mathbf{1}_{\{L_i^s=-Y^s\}}\right]},$$

$$\beta_i = \frac{\sum_{s\in S_L}\left[\mathbf{1}_{\{L_i^s=Y^s\}} + \mathbf{1}_{\{L_i^s=-Y^s\}}\right]}{|S_L|},$$

for each $i \in \{1, \cdots, m\}$. After estimating these parameters, we can infer the true label distribution by the posterior $P_{\boldsymbol{\alpha},\boldsymbol{\beta}}(Y^s|\boldsymbol{L}^s)$ 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.