Nessy: a Neuro-Symbolic System for Label Noise Reduction

Alisa Smirnova, Jie Yang, Dingqi Yang and Philippe Cudre-Mauroux

Abstract—Noisy labels represent one of the key issues in supervised machine learning. Existing work for label noise reduction mainly takes a probabilistic approach that infers true labels from data distributions in low-level feature spaces. Such an approach is not only limited by its capability to learn high-quality data representations, but also by the low predictive power of data distributions in inferring true classes. To address those problems, we introduce Nessy, a neuro-symbolic system that integrates deep probabilistic modeling and symbolic knowledge for label noise reduction. Our deep probabilistic model infers the true classes of data instances with noisy labels by exploiting data distributions in an underlying latent feature representation space. For data instances where inference is not reliable enough, Nessy extracts symbolic rules and ranks them according to several utility metrics. Top-ranking rules are injected into the deep probabilistic model via expectation regularization, i.e., via a posterior regularization term constraining the class distribution in the objective function. In a real deployment over multiple relation extraction tasks, we demonstrate that Nessy is able to significantly improve the state of the art, by 7% accuracy and 10.7% AUC on average.

Index Terms—noise reduction, neuro-symbolic systems, deep probabilistic model, relation extraction, distant supervision

1 Introduction

The success of deep learning models heavily relies on the quality and quantity of labeled training data [1]. Obtaining large amounts of high-quality training data, however, is a long, laborious, and usually costly process. Addressing the label quantity issue, several approaches have been proposed to enable the fast creation of large training sets by exploiting low-quality but easily accessible labeling sources. Typical methods include distant supervision [2], [3], crowdsourcing [4], and automatic data augmentation [5]. The effectiveness of these approaches, in terms of scalability and cost-effectiveness, has been evaluated on a variety of tasks, ranging from information retrieval [4], [6] and extraction [7], [8], [9] to image segmentation and recognition [10], [11]. Those methods, efficient, result in noisy labels for the training data.

In contrast to the growing body of work addressing the label *quantity* issue, little attention has been devoted to the labels *quality*. Due to the lack of transparency and accountability of deep learning models [12], [13], incorrect labels in the training set are generally difficult to identify; consequently, label noise has become a main obstacle for developing, deploying, and improving deep learning models.

Existing work mainly suggest probabilistic methods that leverage data distributions for debugging noisy labels [14], [15], [16]. The basic assumption is that data points distributed close to each other are more likely to have the same label, hence it is possible to infer the true class of an instance from its neighbors. Those methods, however, suffer from two major limitations. First, they model data distributions in *low-level* feature spaces with oversimplified structures (e.g., lexical features such as tokens or patterns). As for most language and vision problems, the low-level distributions learned by these methods are limited compared to the true distributions on the higher-level latent feature representation space that present complex dependencies among features [17]. The second limitation, and probably the most important one, is that all existing methods are data-driven methods that learn statistical patterns from the data only. As a result, the performance of such methods is further limited by the intrinsic predictive power of data distributions for the true classes, a limiting factor of any data-driven method that relies on data distributions for debugging noisy labels.

Deep neural networks are able to learn data representations that encompass complex dependencies among features and provide data distributions that are more effective for the inference of true labels. Those methods, however, are nevertheless limited by the amount of relevant information in the training data for the inference of true classes. The presence of noisy labels makes it even harder for neural networks to recognize such relevant information for truth inference. Besides, deep learning methods—despite their flexibility—are typically not robust to noise in the training data, especially when the size of the training data is not sufficiently large. Compared to data-driven approaches, knowledge-driven approaches are more robust due to their capability in representing concepts and causal relations among them. Those two types of approaches are complementary to each other in the sense that knowledge can be acquired in parallel to the data itself (e.g., from humans) and can be integrated into data-driven methods to improve the effectiveness of models. Recent discussions in the AI communities have converged on the idea of integrating symbolic methods with machine learning, i.e., neuro-symbolic methods, which benefit from the robustness of symbolic methods, the flexibility of machine learning, and their complementarity in utilizing different information sources [18].

Inspired by those developments, we propose to integrate deep learning with symbolic knowledge for label noise reduction. In that way, deep learning methods are used to

infer the true classes from data distributions on the latent feature representation space, and symbolic knowledge is introduced to further improve the accuracy of the inferred labels, in particular for instances for which the latent feature representation learned by the deep neural model is dominated by class-irrelevant information. However, developing such an approach is challenging. The first challenge is to develop a deep learning model that can learn high-quality data representations to capture data distributions while using them for true class inference. Second, it remains unclear what knowledge is the most beneficial for the model, how to identify such knowledge, and how to integrate that knowledge into the deep learning model for noise reduction.

In this paper, we present Nessy, a new neuro-symbolic system that takes advantage of *deep* and *probabilistic* modelling for inferring latent classes, and of *symbolic knowledge* expressed as a set of logic rules for improving model inference. Unlike previous probabilistic methods, our deep probabilistic model adopts deep neural networks to parameterize data distributions, thus is able to model complex feature relationships in the data and to learn high-quality latent feature representations. Nessy then extracts logic rules from the data following the recommendations of the domain experts and injects them into the deep probabilistic model to improve the accuracy of true class inference. To extract the most beneficial rules, Nessy employs a data sampling component that selects instances for which the probabilistic inference is unreliable. Those rules are ranked by several metrics that quantify the potential utility of the rules for label noise reduction, including support (how many instances match the rule) and confidence (how useful the rule is in discriminating one class from another). To inject rules into the deep probabilistic model, Nessy leverages expectation regularization [19] that takes rules as soft constraints. The rules are added to the objective function of the deep probabilistic model to regularize the inference results.

The proposed system is task-independent and can be used for any classification problem. In this paper, we investigate the effectiveness of Nessy on the task of relation extraction. Historically, distant supervision is widely used to obtain training data for this task [3], [20]: some external Knowledge Base is used to obtain the relations between two entities; the relations are then used to automatically label sentences that mention those entities. This approach inevitably brings noise into the training data, i.e., some of the labels are incorrect (see example on Figure 1). We use the term *distant supervision noise* to denote this type of noise. In our experimental evaluation, we demonstrate the effectiveness of Nessy in reducing both the distant supervision noise and random noise, across multiple relations and different noise levels.

In summary, we make the following key contributions:

- We introduce the notion of debugging noisy training data through a neuro-symbolic approach;
- We propose a deep probabilistic model that infers the true classes of training instances with noisy labels by learning their data distributions in the latent representation space;
- We present Nessy, a system architecture that orchestrates the operations of the deep probabilistic model and of the symbolic methods through a set of components for rule extraction, ranking, and integration.
- We demonstrate the effectiveness of our approach through an extensive evaluation on several relation extraction tasks and show that Nessy improves the state of the art by 7% in accuracy and 10.7% in AUC.

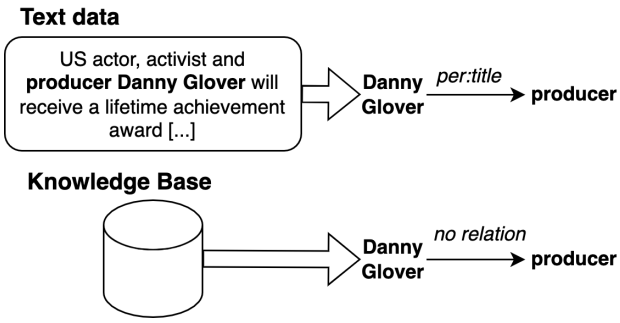


Fig. 1. In the manually labeled data, this instance is labeled with relation `per:title` between the two following entities: **producer** and **Danny Glover**. However, this information might be missing from the Knowledge Base; consequently, this data instance is labeled as not having relation `per:title`.

To the best of our knowledge, we are the first to suggest a neuro-symbolic approach for debugging noisy training data. Compared to our previous work Scalpel-CD [21], where we introduced deep probabilistic modeling coupled with a human-in-the-loop approach for post-processing the model's inference, Nessy introduced in this paper is an alternative approach that involves humans in a much lighter way for improving model inference: humans only need to examine the validity of the rules in Nessy, whereas in Scalpel-CD they need to process data on a per-instance basis. Results show that Nessy outperforms its predecessor on challenging datasets with distant supervision noise by 4.3% in accuracy while requiring much less human input.

2 Related Work

Label Noise Reduction. Existing methods are mainly developed for crowdsourcing and distant supervision. In crowdsourcing, noise reduction has been a central problem as worker annotations are often noisy. Typical methods assume a redundancy of worker annotations, e.g., majority voting and those based on Expectation Maximization (EM) [22]. EM methods simultaneously estimate the true labels and parameters of the annotation process, e.g., worker reliability by Dawid and Skene [23] and task difficulty by Whitehill et al. [24]. Our work is different in that we consider the broader scope of noisy data, where labels are not necessarily redundant.

Distant supervision is a popular approach for creating training data for relation extraction. Variety of methods were proposed to tackle distant supervision noise. Those methods mainly leverage data distributions in the low-level feature space, e.g., the factor graph model by Riedel et al. [14] and the generative model by Takamastu et al. [15]. Alfonseca et al. [16] introduce a hierarchical model to capture the data distribution in the topic space. In comparison, our deep generative model is flexible to capture data distributions with more complex data structures.

Most of the deep learning models designed for distantly supervised relation extraction do not explicitly filter out noisy instances. Instead, attention mechanism is used to weigh the instances depending on their relevance to the relation [25], [26].

Neural-Symbolic Learning. While being flexible to capture complex mapping between the features and label, machine learning approaches – deep learning in particular – are generally data-hungry (sample inefficient) and not robust—they only learn (possibly spurious) statistical correlations. Compared to data-driven, learning-based approaches, knowledge-driven, reasoning-based approaches are more sample-efficient and more robust due to their capability in representing concepts and the causal relations among them. Recent discussions in the AI communities have converged on the idea of integrating symbolic methods with machine learning. A visible trend is the growing body of work on neural-symbolic methods [27]. For instance, Xu et al. [28] introduce the semantic loss that augments the training objective of neural networks with soft-constraints specified with domain knowledge; Allamanis et al. [29] propose to learn continuous representations of symbolic knowledge for integration into neural networks. On the application side, neural-symbolic methods have been applied to both vision and language tasks including visual relation prediction [30], visual question answering (VQA) [31], [32], and semantic parsing [33]. In debugging noisy training data, deep learning approaches suffer more from the sample-efficiency and robustness issues. To the best of our knowledge, we are the first to investigate the integration of symbolic knowledge into deep learning for debugging noisy training data.

We note though, that there are some work making use of symbolic knowledge for distantly supervised relation extraction: Koch et al. [34] filter out relation instances with incompatible entity types; Ji et al. [26] add a constraint to the objective function that is based on the entity descriptions extracted from Wikipedia pages. Rocktäschel et al. [35] explore logical dependencies between relations. However, none of those work focuses on the noise reduction.

3 Architecture

Figure 2 gives an overview of our system. Nessy takes as input a noisy dataset, e.g., in the context of relation extraction (the main application we are currently exploring in the context of this work), the dataset is a corpus of sentences, each associated with a noisy label that represents a relation between two entities. The dataset is first passed to a deep probabilistic model (C1 on Figure 2), which infers the latent true class for each data instance along with a high-level latent feature representation of it. The inferred class is subsequently used as an input to a data sampler (C2), which calculates the reliability of the model's inference and selects data instances that are fed to the knowledge extractor (C3). The knowledge extractor extracts symbolic rules from the sampled instances and ranks them according to rule utility metrics that measure their potential benefit for improving the deep probabilistic model. Then, the top-ranking rules are injected to the model through a knowledge injector using expectation regularization (C4). In this way, the model is integrated with additional knowledge for better predictions. Next, we describe the components in more detail.

C1: Deep Probabilistic Model. The deep probabilistic model receives a noisy dataset and simultaneously infers two types of latent variables, i.e. the latent true class and the latent feature representation. The latent feature representations, represented as a set of low-dimensional vectors, capture the underlying data structure for the inference of the latent true classes. The inference process relies both on existing noisy labels and latent feature representations. The basic idea is that data instances distributed around the same region of the latent feature representation space are likely to belong to the same class. Consider for example a data instance whose surrounding neighbors are all positively labeled; it is then likely that the instance be positive also, even if its existing noisy label is actually negative. In most cases, the surrounding neighbors are partially labeled as positive and partially as negative. The deep probabilistic model is able to strike a balance between the latent features and the noisy labels to obtain a reasonable estimate of the latent class. This component is described in further detail in Section 4.1 and 4.2.

C2: Data Sampler. The data sampler component is critical in identifying the rules that the deep probabilistic model might have missed during training. Its main purpose is selecting data instances for which the deep probabilistic model's inference is not reliable enough. As in previous work [21], the model reliability is approximated by the (inverse) uncertainty of model inference. In addition, it further considers random sampling as an alternative sampling method, which is more suitable for data with structural noise. Additional detail on our sampling algorithm is given in Section 4.3.

C3: Knowledge Extractor. From sampled data instances, Nessy then extracts symbolic knowledge as a set of rules for complementing the deep probabilistic model. In this work, we consider textual features from the input sentences to compose rules, e.g., part-of-speech tags and NER tags. The extracted rules are ranked according to their potential utility in improving the accuracy of label inference by the deep probabilistic model. The utility is approximated by two metrics: 1) support, denoting the number of instances matching the rule, and 2) confidence, denoting the discriminative power of the rule. We explain our extraction algorithm in detail in Section 4.4.

C4: Knowledge Injector. Nessy injects the extracted rules into the training process using expectation regularisation [19]. The underlying idea is that the instances matching a certain rule should follow a different label distribution than the whole data. This distribution might be estimated using validation data with manual labels whenever available. Otherwise, heuristics might be used to estimate the label distribution. Providing a deep probabilistic model with such a set of rules together with their label distribution can significantly improve both the accuracy of true label inference and model interpretability, as we explain in Section 4.5 and demonstrate in our empirical evaluation in Section 5.

4 Nessy Components and Trade-offs

Nessy orchestrates human and machine intelligence via the set of components described in the previous section. These

Fig. 2. The architecture of Nessy: our system takes as input a noisy training dataset and infers the true classes of data instances with deep probabilistic modeling; it identi es from the data instances for which inference is unreliable, and extracts and ranks symbolic rules by their potential utility in improving the accuracy of label inference by the deep probabilistic model. Top-ranking rules are then injected into the deep probabilistic model for model retraining and improvement.

components seek a trade-off between different key features of our system that have impact on the resulting system performance. The deep probabilistic model infers the latent class of a data instances by striking a balance between the trust in existing noisy labels and the data distribution. Similarly, the data sampler component is concerned with the trade-off between data representativeness and label reliability: it aims at selecting data instances that are most representative to capture useful rules, and those whose labels are most unreliable to effectively reduce noise in the training set. The knowledge extractor seeks the rules that are frequent in the data, but are not (yet) captured by the deep probabilistic model. Finally, the knowledge injector strikes a balance between the model's predictions and the knowledge-driven constraints.

In this section, we rst brie y describe our deep probabilistic model, and then characterize the trade-off space for all four system components: deep probabilistic model, data sampler, knowledge extractor, and knowledge injector. For each of them, we propose an algorithmic solution that simpli es the search of the optimal trade-off.

4.1 Deep Probabilistic Model

Overview. Our goal is to infer a true latent class label given a data instance and its noisy label. To infer the latent class of a data instance, the basic intuition of our deep probabilistic model is that data instances similar to each other (i.e., close in the latent feature space) are more likely to belong to the same class. With this in mind, we utilize generative modeling to model the relation between observed variables (i.e., data instance and its noisy label) and latent variables (i.e., latent features and latent true class). Parameters of the generative model are learned simultaneously with the inference of true labels given a dataset with noisy labels. In the following, we rst describe the generative model and then the inference and learning processes.

Generative model. We consider a generative model to fully capture the data distribution of a large training dataset. Formally, the process is described as follows. Given a noisy dataset $D = \{(x_i, \hat{y}_i)\}_{i=1}^{N}$, where x_i is a data instance and $\hat{y}$ is its corresponding noisy label (we omit the index i whenever it is clear that we are referring to a single data instance). For each data instance $(x_i, \hat{y}_i) \in D$:

Draw a latent feature vector $z_i \sim P(z)$ where $P(z) = \mathcal{N}(0; I)$ is a standard Gaussian distribution;
Draw a latent class $y_i \sim P(y)$ where $P(y) = \mathrm{Cat}(y|\ )$ is a Multinoulli distribution, where $= \{1/K; \ldots; 1/K\}^{|}$ (K is the number of classes). Multinoulli (or Categorical) distribution suits best for encoding a discrete random variable;
Draw a data instance $x_i \sim P(x|z; y)$;
Draw a noisy label $\hat{y} \sim P(\hat{y}|z; y)$.

Our generative model is depicted in Figure 3. The data instances and the noisy labels are both dependent on the latent feature vector z and on the latent class y, which captures the class speci cation. z can include both class and non-class related features. For example, for a sentence x, the latent feature vector z can represent topics related to a certain class, and additionally, it can capture the author's writing style that is not class-related. The non-class related features, when mistaken as class-related, are likely to lead to wrong labels during the labeling process. z and y are conditionally dependent given the observed data instance x and the noisy label $\hat{y}$.

Formally, the deep probabilistic model is expressed by the following factorization:

$$P(x; \hat{y}; z; y|\ ;\ ) = P\ (x|z; y)P\ (\hat{y}|z; y)P(z)P(y); \qquad (1)$$

The likelihood functions $P\ (x|z; y)$ and $P\ (\hat{y}|z; y)$ are parameterized by deep neural networks to accurately capture the distributions of the data and the noisy labels, and and denote the set of parameters of the corresponding networks. Depending on the speci c form of the data, different likelihood functions can be used for $P\ (x|z; y)$: Gaussian likelihood is suitable for image data, while Multinomial likelihood is more suitable for textual data (see [36], [37]). $P\ (\hat{y}|z; y)$ is represented by a Multinoulli likelihood.

Inference and learning. The inference for the latent feature vector z and the latent class y is closely related to the learning of the deep probabilistic model parameters. The parameters are learned by maximizing the log likelihood of the observed data instances and the associated noisy labels:

$$\log P(x; \hat{y}|\ ;\ ) = \log \int\int P(x; \hat{y}; z; y|\ ;\ )\,dz\,dy: \qquad (2)$$

Due to the intractability of the integral, we approximate the true posteriors of z and y with variational ones, denoted by $Q\ (z|x; y)$ and $Q\ (y|x)$, respectively. We use a Gaussian

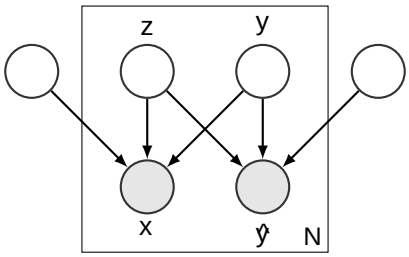


Fig. 3. Graphical model of the generative model. The data instance x and the noisy label $\tilde{y}$ are both generated from distributions conditioned on the latent feature vector z and the latent class y. θ and ϕ are the parameters of the likelihood function for generating the data and noisy labels, respectively (the priors of z and y are omitted). x and $\tilde{y}$ are observed variables; all the rest is inferred or learned.

distribution for $Q_\theta(z|x,y)$ and a Multinoulli distribution for $Q_\phi(y|x)$. Given the complex dependencies among the low-level features of data instances and their relationships with the latent variables, these distributions are again parameterized by deep neural networks. The parameters, including those of the generation networks θ and ϕ and those of the inference networks θ and ϕ, are then learned by maximizing the evidence lower bound (ELBO) [38] of the objective as follows:

$$\begin{aligned}\mathcal{L} = {} & \mathbb{E}_{Q_\theta(z,y|x)}[\log P_\phi(x,\tilde{y}|z,y)] \\ & - D_{KL}[Q_\theta(z,y|x) \| P(z,y)],\end{aligned} \tag{3}$$

where $\mathbb{E}(\cdot)$ is an expectation and $D_{KL}[\cdot\|\cdot]$ is the KL-divergence between two distributions. The first term in Eq. 3 corresponds to the generative model that aims to reconstruct the data instances and their labels (reconstruction error). The second term corresponds to the inference model and can be seen as a regularization term to prevent over fitting. [1]

4.2 Inferring the Latent Class

From a neural network perspective, the first part of the ELBO (Equation 3) can be interpreted as the sum of the (negative) reconstruction errors of the observed feature x and noisy label $\tilde{y}$. Hence, we rely both on data distribution and existing noisy labels when inferring latent classes.

The original ELBO weights both types of reconstruction errors as equally important. While this can result in a powerful generative model for data and label generation, our goal is different: we are interested in inferring latent classes, for which data distribution and existing noisy labels might have different importance. Furthermore, their relative importance varies across different datasets: for datasets whose data distributions are highly indicative of the true class and for which the noise ratio of the existing labels is high, the inference process should rely on the data distribution more heavily. Otherwise, existing labels should be trusted more.

Therefore, it is natural to extend the original ELBO by introducing a parameter α to weight the importance of the two types of reconstruction errors:

$$\begin{aligned}&\mathbb{E}_{Q_\theta(z,y|x)}[\log P_\phi(x,\tilde{y}|z,y)] = \\ &\mathbb{E}_{Q_\theta(z,y|x)}[\log P_\phi(\tilde{y}|z,y)] + \alpha\,\mathbb{E}_{Q_\theta(z,y|x)}[\log P_\phi(x|z,y)].\end{aligned} \tag{4}$$

1. We describe the detailed steps for deriving the ELBO in the supplementary material.

Algorithm 1: Learning Deep Probabilistic Model

Input: the set of N i.i.d. data instances $\mathcal{D} = \{x_i, \tilde{y}_i\}_{i=1}^{N}$, ELBO adapter α, and the maximum number of iterations $Iter$

1 Initialize θ; ϕ; α;
2 **for** $t = 1; t \le Iter; t{+}{+}$ **do**
3 Sample a batch of data instances;
4 **forall** x_i in the batch **do**
5 Compute y_i and z_i;
6 Compute the noisy gradient ∇_θ; ∇_ϕ; $\nabla_\alpha \mathcal{L}$;
7 Average noisy gradients from batch;
8 Update θ; ϕ; α with gradient descent;
9 **if** $\mathcal{L}$ has converged **then**
10 break;

In case $\alpha \neq 1$, we are no longer optimizing the ELBO on the log marginal likelihood (Equation 2). When $\alpha < 1$, we put more weight on existing labels rather than the data distribution; consequently, the model will tend to refrain from denoising labels by resorting to the data distribution. Otherwise, when $\alpha > 1$, we are weakening our trust on the existing noisy labels while putting more weight on the data distribution in determining label noise. While this allows for more flexibility, it could also bring additional noise to the labels due to the non-class related features in the data distribution. Identifying a proper value of α is important to achieve a good performance in label noise reduction.

The resulting optimization algorithm is given in Algorithm 1. It iteratively goes over two steps, i.e., the forward and the backward step. At each iteration, the forward step (row 5) computes the latent variables given current parameters; the backward step (row 6-8) then updates the parameters by backpropagating the gradients of the errors. In the calculation of the gradients (row 6), we use an adapted version of the ELBO (Equation 4).

4.3 Data Sampling

New symbolic rules that can best improve the inference accuracy of the deep probabilistic model are likely to rise from instances where our deep probabilistic model's inference is most unreliable. We therefore extract rules from such instances. The model reliability is approximated by the inverse of the model's uncertainty, measured by Shannon entropy [39], [40]:

$$H[y|x] = \sum_{C=1}^{K} p(y = C|x) \log p(y = C|x) \tag{5}$$

where C is the class and K is the number of classes. We hypothesize that uncertainty sampling is effective for noisy datasets where labels might be disconnected from the data (e.g., random noise, given that the noise ratio is lower than $1/K$). In this case, for data instances far away from the true decision boundary, the deep probabilistic model can take advantage of the majority label of the data instances and provide satisfactory results; however, for data instances close to the decision boundary, the deep probabilistic model will have a high uncertainty due to the mix of data instances from different classes and might generate incorrect results.

TABLE 1
Example of the rules extracted from one instance. Subject and object entities are boldfaced ("Dillinger" and "director", respectively).

Sentence	Extracted Rules
Folded into **Dillinger** 's of ce , the program is headed by director **Duncan McCormack** , a 51-year-old with a diverse background that includes mental health counseling .	"SUBJ-PERSON 's of ce , the program is headed by OBJ-TITLE", "SUBJ-PERSON POS NN , DT NN VBZ VBN IN OBJ-TITLE", "SUBJ_LEFT_O", "OBJ_RIGHT_PERSON"

For datasets with structural noise (i.e., when labels are generated from data as in [41]), there can exist substantial regions of the data distribution where the majority of the labels are incorrect. In that case, the inference of the deep probabilistic model can be totally off while being highly probable according to the model. Therefore, we also consider random sampling, which is independent of the deep probabilistic model. The effectiveness of the two sampling methods can be evaluated using a set of validation instances with ground truth labels; in this context, a good sampling method is one from which the sampled data instances cover more validation instances where the deep probabilistic model's inference is wrong.

4.4 Knowledge Extraction

Knowledge extraction focuses on extracting rules from sampled data instances. We consider rules where the rule body contains features and its head indicates the class of instances matching the rule. Features composing the rules depend on the domain knowledge of speci c tasks. In the context of relation extraction tasks, we consider the following features:

- The sequence of words between the two entities;
- The part-of-speech tags of these words;
- The NER tag of the word to the left of the left-most entity;
- The NER tag of the word to the right of the right-most entity.

We give an example of such rules in Table 1. This set of rules is similar to the lexical features used by Mintz et al. [3], which proved their ef ciency on relation extraction from a distantly supervised dataset.

After the rules are identi ed, we select the ones that are potentially useful for label denoising. The quality of the rules with respect to noise reduction depends on two important factors: *support* and *con dence*. Rule support is de ned as the number of instances that match the rule and corresponds to the rule's coverage. Con dence of the rule corresponds to the rule's effectiveness in discriminating one class from another; it is de ned as the percentage of instances supporting the rule that have the same label. Intuitively, both rule support and con dence should be high for the rule to be effective in improving label inference.

The noise ratio of the dataset also plays an important role in identifying rules that might be useful for label noise reduction. The noisy labels have an average accuracy $1 - NR$ where NR denotes noise ratio. Suppose that we are using a rule alone to predict a label (i.e., we assign the most frequent label to all data instances supporting the rule). Let S denote rule support, $Conf$ denote rule con dence, and $N = S \cdot Conf$ is the number of instances that support the rule and have the same label. The number of correct predictions is a sum of true positive and true negative predictions. Therefore, accuracy of the rule is de ned as follows:

$$\frac{\overbrace{N \cdot Conf}^{\text{true positives}} + \overbrace{(S - N) \cdot (1 - Conf)}^{\text{true negatives}}}{S} = Conf^2 + (1 - Conf)^2 \quad 1 - NR; \tag{6}$$

By solving this inequality, we obtain the following restriction:

$$Conf \quad \frac{1}{2} + \sqrt{\frac{1}{4} - \frac{NR}{2}} \tag{7}$$

That is, rule con dence should be high enough. For example, if rule support is 1000 and the noise ratio is 20% then the number of instances of the same class conforming to that rule should be at least 888 for a rule to be ef cient (i.e., rule con dence 0:888).

The full algorithm for extracting rules is given in Algorithm 2. The rules identi ed as potentially useful for label denoising are then passed to the knowledge injector.

Algorithm 2: Extracting the Rules

Input: Data instances represented by latent features and inferred classes $D = \{z_i; P(y_i)\}_{i=1}^{N}$, validation instances with ground truth labels V, sampling ratio p
Output: Set of rules R

```
R ← ∅;
Decide the sampling strategy using V;
D̂ ← Pick p top-ranking data instances
foreach d_i ∈ D̂ do
    R_i ← rules extracted from d_i
    R ← R ∪ R_i
R ← Pick the rules with the highest support
R ← Keep only the rules that satisfy Eq. 7
```

4.5 Knowledge Injection

The key idea behind Nessy is to strike a balance between the labels given by a set of symbolic rules and predictions inferred from the data. To inject the rules into the model, we apply expectation regularization [19]. An additional term is added to the loss function that promotes model predictions on a subset of the data instances to match a rule-speci c expectation: the likelihood of an instance that matches the rule belonging to a class. This expectation might either be some human prior knowledge or it can be estimated using the labeled data (e.g., development set). Formally speaking, let $\hat{p}$ be the model predictions and p be the provided expectation, then the expectation regularization term for the objective function is

$$(p; \hat{p});$$

where $$ is a distance function, e.g., KL-divergence as we use in this work.

Recall Equation 3: the loss function of our deep probabilistic model has a regularization term expressed as the KL-divergence between the posterior and prior distributions of

Algorithm 3: Injecting the Rules

Input: Raw data instances D, set of rules R, validation instances with ground truth labels V
Output: Batch generator

```
foreach r_i ∈ R do
    P̂_i ← Estimate label distribution using V
    R_i ← Collect data instances from D that match r_i
forall R_i do
    shuffle R_i
    yield micro-batches from R_i, P̂_i
```

z and y. Therefore, the additional constraints on the distribution of the latent variable y can be naturally expressed using the existing regularization term. To do this, we first factor the second term in Equation 3 into two parts:

$$D_{KL}\left[Q_{;}(z;y|x)\|P(z;y)\right] = D_{KL}\left[P(y|x)\|P(y)\right] + D_{KL}\left[P(z|x)\|\mathcal{N}(0;I)\right] \quad (8)$$

By default, $P(y)$ is a Multinoulli distribution as explained in Section 4.1. Our goal is to predict a true label y, therefore, we add knowledge-driven constraints into the corresponding regularization term from Eq. 8. We encourage our deep probabilistic model to match the posterior distribution $\hat{P}(y|r_i(x))$ if data instance x matches rule r_i (we denote it as $r_i(x)$). In addition, we introduce a coefficient for the expectation regularization term. Similarly to , the parameter reflects the strength of our beliefs about the provided label expectation.

$$D_{KL}\left[Q_{;}(z;y|x)\|P(z;y)\right] = \sum_{r_i \in R} D_{KL}\left[P(y|x)\|\hat{P}(y|r_i(x))\right] + D_{KL}\left[P(z|x)\|\mathcal{N}(0;I)\right] \quad (9)$$

To train the model with a set of knowledge-driven constraints, we change the micro-batch generation in a way that each batch contains only the instances that match the same rule. We verify that each instance appears only once during one training epoch. The algorithm for micro-batch generation is given in Algorithm 3. The generated batches are used in the mini-batch optimization algorithm (Algorithm 1) to retrain the parameters of the deep probabilistic model.

5 Experiments and Results

In this section, we present experimental results for evaluating the performance of Nessy on several relation extraction tasks with two types of noise: random noise and distant supervision noise. We start by presenting an evaluation of the key components of our system by answering the following questions:

- Q1: How well does the deep probabilistic model perform when inferring the latent class from noisy data?
- Q2: How well do the uncertainty and random sampling perform on datasets with different types of noise?
- Q3: How does the impact of the rules depend on their utility properties?
- Q4: How effective is knowledge injection for improving model performance in label inference?

For each of the above questions, we also analyze the influence of the type of noise.

TABLE 2
Characteristics of the datasets with distant supervision noise.

Dataset	# Train instances	# Test instances	Noise ratio
Title	4621	974	0.418
Employee	2821	604	0.382
Top Members	7071	1258	0.281

5.1 Experimental Setup

Datasets. We evaluate Nessy on the TAC Relation Extraction Dataset constructed by Zhang et al. [42] from the TAC KBP evaluations (2009-2015) and annotated by crowds. Since this dataset includes the ground truth, it allows us to fairly evaluate our proposed noise reduction methods [2]. Moreover, when knowing gold labels we can model different types of noise and analyze their properties. Given the original TACRED, we sub-sample the datasets for binary classification for three relation labels that have the maximum number of instances. These relation labels are per:title (Title dataset), org:top_members/employees (Top Members dataset) and per:employee_of (Employee dataset). We took all positive instances for each dataset, and for the negative ones, we sub-sample the instances that contain entities with types compatible with a given relation. Then, we add noise into each dataset. We consider two types of noise for each dataset: distant supervision (DS) noise and random noise. To create DS noise, we apply the original pipeline proposed by Mintz et al. in [3]. To that end, we perform entity linking to Wikipedia articles using the BLINK software [43] for entities of type Person (in all three datasets, either the subject or object entity is of type PERSON). To obtain the labels, we perform an exact match of the second entity with the corresponding Wikipedia abstract. The noise ratio varies between the datasets (see Table 2).

For random noise, we fix the noise ratio similar to the one for distant supervision noise by flipping the relation label with a probability equal to that noise ratio.

Noise Type Analysis. Distant supervision noise is structural (i.e. not completely random) and depends on the procedure used to create the labels, while random noise is uniformly distributed across the labels. TACRED is derived from news wires and online text, therefore, the sentences might mention entities that are not very popular, and thus, are missing from the Knowledge Base (this problem is known as Knowledge Base incompleteness). This situation is more likely than the opposite, when there is a relation instance in the Knowledge Base but the sentence does not express a relation between the given entities. This causes some label distribution shift, which we cannot avoid unless we subsample the datasets. While in the Title dataset 52.9% of instances are positive (according to the gold labels), the noisy datasets contain 50.5% and 36.1% of positive instances for random and distant supervision noise, respectively. A similar observation holds for the other two datasets. For the Top Members dataset, the percentage of positive instances drops from 26.7% to 17.1% while for the Employee dataset

2. Note that in real-life scenario noise ratio might be unknown. Noise ratio can be estimated using reasonable heuristics from domain experts or by labeling a representative subsample of the data.

it drops from 54% to 25.6%, making the distant supervision noise datasets more challenging than random noise ones.

Comparison Methods. We compare the following data-driven noise reduction methods to our system. 1) Ratio [44]: a ratio-based method that nds the most predictive features and identi es data instances with the most uncertain labels as those containing such features yet labeled differently from the label indicated by the features. The predictive power of a feature is calculated as the ratio between the number of data instances of a certain class containing such a feature and the overall number of data instances. 2) Pattern [15]: a generative model designed to capture the labeling process as a generative process from the latent classes. The model facilitates the inference of the posterior of the latent class given the observed features, thus it can be used for noise reduction. However, unlike our deep probabilistic model that infers the latent class by exploiting the data distribution in the latent feature space, Pattern is a probabilistic model that directly models the generative process of low-level features. 3) HierTopic [16]: a hierarchical topic model that assumes a latent topic-word hierarchy for the data generation process. Unlike our probabilistic model which learns complex hidden data structures, HierTopic is only capable of learning a hierarchical structure. 4) Scalpel-CD [21], our previous approach that involves human workers for debugging noisy labels. For our proposed system, we compare an automated variant and the full system incorporating symbolic knowledge: 5) DPM , our proposed deep probabilistic model, which is used in isolation to automatically correct wrong labels. 6) Nessy, our proposed system which makes use of both deep probabilistic modeling and symbolic knowledge.

Parameter Settings and Model Training. We empirically set optimal parameters based on the development set. These include hyperparameters for the deep probabilistic model architecture, model con guration, i.e., values of and and those for model training, e.g., batch size and learning rate. As an input for the inference network of the deep probabilistic model Q and Q we use the sentence representations inferred by PA-LSTM model [42] that is trained on the same noisy dataset with default hyperparameters. Doing so allows us to have feature-rich sentence representations while avoiding extensive model engineering that is usually required for such complex tasks. It is worth noting that these sentence representations also carry some label noise. Additional details of the hyperparameters for both the neural architecture and training are included in the supplementary material.

Evaluation Protocols. We separately evaluate the performance of Nessy on noise reduction and its effect on relation extraction using different protocols. For noise reduction, we train the model on the noisy training dataset and compare our predictions on the training set with the ground truth. For relation extraction, noise reduction by our system is performed on the training set. A PA-LSTM model [42] is then trained on the denoised training set and evaluated on the test set.

We measure the performance of our proposed system in noise reduction using two metrics: accuracy and Area Under the ROC Curve (AUC). Note that a higher ROC AUC indicates a greater distinction between True Positives and True Negatives and hence better latent representations.

(a) Validation Accuracy (b) Validation AUC

(c) Loss on Label Reconstruction (d) Loss on Data Reconstruction

Fig. 4. Comparison of performance against loss in data and label reconstruction with different settings of on TACRED with 40% random noise. The upper gures represent (a) the accuracy and (b) AUC on the validation set during the training process; the lower gures depict the reconstruction loss (negative log-likelihood) of noisy labels (c) and data instances (d).

5.2 Inferring the Latent Class (Q1)

To understand the effectiveness of our deep probabilistic model in inferring the latent class, we analyze the relationship between capturing the data distribution and inferring the latent class. We compare the performance of the deep probabilistic model with different settings of , which controls the relative importance of the reconstruction errors of data instances and that of the noisy labels in the objective function. We analyze the dynamics of our deep probabilistic model on the validation set during the training process, with selected from $\{0.001, 0.01, 0.1, 1, 10\}$. Results on the Title dataset with random noise and with a noise ratio $NR = 40\%$ are shown in Figure 4.

From Figures 4a and 4b, we observe that with the increase of , the performance of the deep probabilistic model rst increases but then decreases. values in a range from 0.01 to 1 yield better results, with the best performance achieved with $= 1$. Such a result is aligned with our previous work [21], con rming the need for striking a balance in minimizing the two different types of reconstruction errors (data and noisy labels). This can be further veri ed by Figure 4c and Figure 4d. We observe that the label reconstruction error decreases fast for all values of , whereas there is signi cant difference in the decreasing speed of the data reconstruction error; the relative importance of those two types of errors, therefore, plays an important role in the inference of true classes. We note that the similar decreasing speed of the label reconstruction error is due to the pre-trained sentence representations already trained on the noisy labels.

Impact of Datasets and Noise Ratios. We observe similar results for all other datasets. The optimal value of is between 0.01 and 1 and performance on $= 1$ is slightly higher than that on other values. Experiments with different noise ratios for the Title dataset show that for $NR = 30, 40\%$ the model

achieves performance higher than 1 NR. However, with lower noise ratio (NR = 20%) the performance is slightly worse yielding an accuracy 79.8%. Overall, we observe that relative performance Acc: (1 NR) monotonically increases with the increase of noise ratio. This means that using the deep probabilistic model is more ef cient for larger noise ratios while for lower noise ratios its effectiveness is limited by the predictive power of sentence representation.

5.3 Uncertainty and Random Sampling (Q2)

The data sampling component of Nessy allows for two sampling strategies, i.e., uncertainty sampling and random sampling, which are model-dependent and model-independent, respectively. We argue that uncertainty sampling is more effective for identifying wrong predictions for both random and distant supervision noise than random sampling [3]. To verify this, we conduct a comparative analysis on each dataset with both types of noise. The comparison is carried out by 1) visually locating in the latent feature space the selected data instances using the two sampling strategies and the data instances where the inference of the latent classes is wrong, and 2) statistically comparing the coverage of the selected data instances on data instances where the inference is wrong.

We start by investigating the effectiveness of uncertainty sampling on the Title dataset with random noise (noise ratio 40%) and DS noise that has similar noise ratio. We apply t-SNE [45] to embed the latent features of the data instances into a two-dimensional space, and then visualize the noisy label and the inferred latent class in Figures 5a, 5b and Figures 6a, 6b. Given our previous results (high performance of our deep probabilistic model in latent class inference in Figures 4a and 4b), we observe from Figures 5a, 5b that the inferred latent features are highly indicative of the latent class: data instances of different classes are located separately in the latent feature space. Figure 5c then shows the data instances for which the inference is correct and incorrect using two colors. We observe that our model can effectively recover the true classes of data instances located far away from the model's decision boundary. In contrast, the majority of the data instances with wrong class inference are located close to the decision boundary. This is due to the fact that at the decision boundary, data instances of different classes are mixed with each other, leading to high model uncertainty. This is con rmed by Figure 5d, which visualizes the top-20% data instances selected by uncertainty sampling. Comparing Figures 5c and 5d, we observe that uncertainty sampling is a highly effective method in selecting data instances when the model inference is wrong. This result is further veri ed by Figure 7a, where we observe that the instances selected by uncertainty sampling show signi cantly higher coverage for instances where the inference is wrong.

Impact of Noise Types. We observe similar results with DS noise (see Figure 6). This con rms our intuition that distant supervision noise is similar to random noise as for both types of noise, the labels are disconnected from textual data. However, comparing Figures 7a and 7b, we observe that uncertainty sampling is slightly less effective in case of DS noise than for random noise. For the Employee dataset with DS noise, we also observe high coverage of uncertainty sampling: 20% of the selected instances cover 50% of the wrong predictions. The Top Members dataset is more challenging due to a higher class imbalance (only 26.7% of the original labels are positive) causing a fuzzy decision boundary between the two classes; consequently, uncertainty sampling performs equivalently well to random. Overall, our evaluation shows that for both DS and random noise, uncertainty sampling performs at least as well as random sampling while achieving notably higher performance most of the time.

5.4 Impact of Symbolic Rules (Q3)

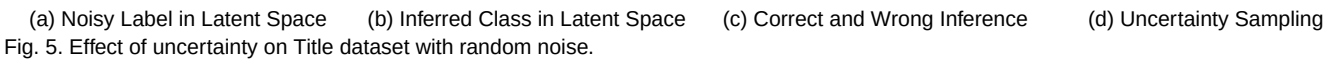
For all datasets, we automatically extract rules according to the instances sampled by uncertainty sampling with ratio $p = 0:2$. For each of the extracted rules, we plot the support value (i.e., the number of instances matching a certain rule) against the number of sampled instances in Figure 8. This gives us a better understanding of what rules could not be captured by the model because of noisy labels. Speci cally, a decreasing curve means that the deep probabilistic model is highly uncertain about the instances matching the rule and vice versa. Therefore, rules corresponding to a decreasing curves are considered as the most bene cial for noise reduction. For instance, from Figure 8a we observe that for the Title dataset with random noise, the curves corresponding to the rules “Obj-Title Subj-Person” and “Obj_Right_Person” are slightly increasing, which means that the model captured them during initial training. However, for the Title dataset with DS noise, the curve “Obj_Right_Person” is slightly decreasing (see Figure 8b), therefore, the model is likely to bene t from it.

For the other two datasets, the picture is more precise: there are certain rules that correspond to sharply decreasing curves, such as the “Obj-Person , JJ NN IN DT Subj-Org” rule for Top Members. It improves the resulting accuracy by 2.4% and 3.9% AUC while there are only 68 instances in the dataset matching this rule. Similarly, for Employee the rule “Subj-Person , NN IN DT Obj-Org” with 39 data instances boosts the accuracy by 12.3% and 6.5% AUC. While having the same con dence close to 1, the rule “Obj-Org NN Subj-Person” (see Figure 8d) is less bene cial: it improves accuracy by 6.7% and 4.1% AUC while its support is much higher (128 data instances). Thus, we conclude that uncertainty sampling helps identifying the most useful rules for debugging noisy labels.

It is worth noting that using part-of-speech tags in addition to tokens (see Section 4.4) allows to better generalize over the existing rules and extract rules with higher support.

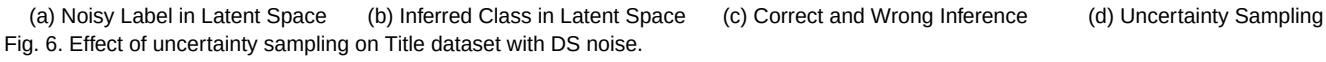
Impact of Noise Types. Table 3 presents the performance of these rules on debugging noisy labels for both types of noise. We use the same value of that yielded the top results of the deep probabilistic model and we select the optimal parameter from $f0:1; 1; 10g$ for each of the rules using the validation set: = 10 for the two rules with the highest coverage and = 1 for the remaining rules. It is worth noting that a rule with low con dence (e.g., “Obj-Title , Subj-Person” with con dence 0:76) might hurt the performance even though is satis es Eq. (7).

3. We have demonstrated the effectiveness of random sampling for structural noise in our prior work [21].



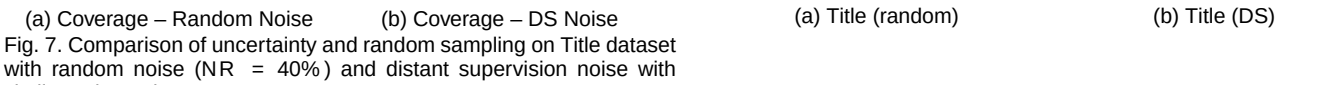
(a) Noisy Label in Latent Space (b) Inferred Class in Latent Space (c) Correct and Wrong Inference (d) Uncertainty Sampling

Fig. 5. Effect of uncertainty on Title dataset with random noise.



(a) Noisy Label in Latent Space (b) Inferred Class in Latent Space (c) Correct and Wrong Inference (d) Uncertainty Sampling

Fig. 6. Effect of uncertainty sampling on Title dataset with DS noise.



(a) Coverage – Random Noise (b) Coverage – DS Noise

Fig. 7. Comparison of uncertainty and random sampling on Title dataset with random noise (NR = 40%) and distant supervision noise with similar noise ratio.

TABLE 3
Performance comparison of different rules on debugging noisy labels on Title dataset for random and DS noise.

Patterns	Random Noise		DS noise	
	Acc.	AUC	Acc.	AUC
Original labels	0.602	0.602	0.582	0.59
No rules	0.713	0.778	0.685	0.743
Obj-Title Subj-Person	0.705	0.770	0.706	0.778
Obj_Right_Person	0.697	0.764	0.687	0.758
Subj-Person, Obj-Title	0.688	0.764	0.636	0.707
Subj-Person, DT Obj-Title	0.706	0.773	0.674	0.742
Obj-Title , Subj-Person	0.691	0.758	–	–
Subj-Person, DT JJ Obj-Title	–	–	0.685	0.763
All rules	0.707	0.775	0.719	0.793

For random noise, we observe that the deep probabilistic model reconstructs the true labels effectively during the initial training. Using rules does not have signi cant impact on the performance, which is consistent with our conclusions from Figure 8a. DS noise is more challenging for the model. Injecting the rules boosts model performance by 2.4% in accuracy and 3.8% in AUC and allows the model to infer true classes more effectively.

(a) Title (random) (b) Title (DS)

(c) Top Members (DS) (d) Employee (DS)

Fig. 8. Rule support depending on the sampling ratio.

5.5 Performance on Debugging Labels (Q4)

We now analyze the effect of injecting knowledge into the deep probabilistic model and compare performance of our proposed system with state-of-the-art noise reduction models on datasets with DS noise. Table 4 (“Noise Reduction”) presents the experimental results of our deep probabilistic model alone and Nessy as a whole system, along with the baselines including our previous work Scalpel-CD [4]. For Nessy we report the best results achieved with the combination of the rules obtained as described in Section 4.4.

Ratio works well on Title and Employee, however, it fails for Top Members due to its class imbalance and the larger number of features (i.e., patterns between the pairs of entities). Pattern improves label quality for Title and Top Members, however it performs marginally worse than Ratio on Employee. Recall that both methods use low-level features (i.e., patterns between the entities) for label inference.

4. We also include a case study in the supplementary material.

TABLE 4
Performance of the compared methods on two settings: debugging noisy labels and relation extraction. The best performance for each dataset in each setting is boldfaced.

	Methods	Title		Employee		Top Members	
		Acc.	AUC	Acc.	AUC	Acc.	AUC
Noise Reduction	Original labels	0.582	0.59	0.618	0.639	0.719	0.584
	Ratio-based	0.663	0.666	0.627	0.647	0.714	0.571
	Pattern	0.692	0.693	0.624	0.644	0.727	0.637
	HierTopic	0.706	0.678	0.801	0.849	0.751	0.714
	DPM	0.685	0.743	0.849	0.931	0.738	0.706
	Scalpel-CD	0.709	0.713	0.882	0.885	0.747	0.635
	Nessy	**0.719**	**0.793**	**0.973**	**0.997**	**0.776**	**0.788**
Learning	PA-LSTM	0.578	0.626	0.740	0.830	0.736	0.737
	W / DPM	0.691	0.747	0.919	0.994	0.699	0.729
	W / Nessy	**0.736**	**0.806**	**0.998**	**0.999**	**0.765**	**0.817**

Both Ratio and Pattern improve label quality, indicating that even such features carry valuable information about the true label and consequently are useful for debugging noisy labels. On the other hand, HierTopic builds a hierarchical structure from the data and does not improve over Pattern on the Title dataset, but demonstrates its ability to effectively recover true labels on Employee and Top Members. Thus, we conclude that low-level features and modeling the inherent structure of the data are complementary to each other and applying them jointly lies at the core of a robust method for label noise reduction.

Our proposed deep probabilistic model also performs differently on the different datasets. Similar to HierTopic, it is less accurate than Pattern on Title, highlighting the fact that low-level features in this case are indicative enough to achieve a notable improvement over the noisy labels. Our model outperforms HierTopic on Employee and achieves similar performance on Top Members, highlighting the importance of learning data structures for those datasets where low-level features are insufficient.

Nessy achieves the best performance on all studied datasets. Nessy outperforms the deep probabilistic model alone by 6.5% in accuracy and 6.6% in AUC on average. This highlights the importance of integrating symbolic knowledge with deep learning models. Compared to the state of the art, Nessy improves accuracy by 7% and AUC by 10.7% on average on the datasets with distant supervision noise.

Impact on Learning Task. We separately evaluate the impact of using Nessy for denoising training data and for relation extraction (see Table 4, "Learning"). The performance of the state-of-the-art models improves significantly when using our deep probabilistic model on all datasets except Top Members. The model achieves further improvement on each dataset when the whole system is applied. On average, performance of PA-LSTM on relation extraction improves by 14.8% accuracy and 14.3% AUC using Nessy. These results clearly highlight the benefits of debugging noisy labels and the effectiveness of Nessy.

6 Conclusion and Future Work

In this paper, we presented Nessy, a neuro-symbolic system that integrates deep probabilistic modeling with symbolic knowledge for debugging noisy labels. Nessy extracts symbolic rules, ranks them according to their utility, and injects them into a deep probabilistic model via expectation regularization by adding a posterior regularization term to the objective function for constraining model inference on instances that match those rules. Our extensive evaluation on several relation extraction tasks demonstrate that the symbolic rules provide an efficient boost to the deep probabilistic model in inferring the true classes. Integrated with those rules, Nessy significantly improves both label quality and the performance of the state-of-the-art relation extraction models. We make our code available at https://github.com/eXascaleInfolab/Nessy_RE. As future work, we plan to explore possible relationships between the rules and other features [46].

Acknowledgments

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement 683253/GraphInt) and from the Science and Technology Development Fund, Macau SAR (SKL-IOTSC-2021-2023).

References

[1] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *ICLR*, 2016.
[2] R. Bunescu and R. Mooney, "Learning to extract relations from the web using minimal supervision," in *ACL*, 2007, pp. 576–583.
[3] M. Mintz, S. Bills, R. Snow, and D. Jurafsky, "Distant supervision for relation extraction without labeled data," in *ACL*, 2009, pp. 1003–1011.
[4] J. Yang, T. Drake, A. Damianou, and Y. Maarek, "Leveraging crowdsourcing data for deep active learning - an application: learning intents in alexa," in *WWW*. International World Wide Web Conferences Steering Committee, 2018, pp. 23–32.
[5] S. Mesbah, J. Yang, R.-J. Sips, M. V. Torre, C. Lofi, A. Bozzon, and G.-J. Houben, "Training data augmentation for detecting adverse drug reactions in user-generated content," in *EMNLP*, 2019, pp. 2349–2359.
[6] M. Dehghani, H. Zamani, A. Severyn, J. Kamps, and W. B. Croft, "Neural ranking models with weak supervision," in *SIGIR*. ACM, 2017, pp. 65–74.
[7] X. Ren, Z. Wu, W. He, M. Qu, C. R. Voss, H. Ji, T. F. Abdelzaher, and J. Han, "Cotype: Joint extraction of typed entities and relations with knowledge bases," in *WWW*, 2017, pp. 1015–1024.
[8] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N Project Report, Stanford*, vol. 1, no. 12, 2009.
[9] A. Ritter, E. Wright, W. Casey, and T. Mitchell, "Weakly supervised extraction of computer security events from twitter," in *WWW*, 2015, pp. 896–905.
[10] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei, "What's the point: Semantic segmentation with point supervision," in *ECCV*. Springer, 2016, pp. 549–565.
[11] L.-J. Li and L. Fei-Fei, "Optimol: automatic online picture collection via incremental model learning," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 147–168, 2010.
[12] A. Balayn, P. Soilis, C. Lofi, J. Yang, and A. Bozzon, "What do you mean? interpreting image classification with crowdsourced concept extraction and analysis," in *WWW*, 2021, pp. 1937–1948.
[13] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.
[14] S. Riedel, L. Yao, and A. McCallum, "Modeling relations and their mentions without labeled text," in *ECML-PKDD*. Springer, 2010, pp. 148–163.
[15] S. Takamatsu, I. Sato, and H. Nakagawa, "Reducing wrong labels in distant supervision for relation extraction," in *ACL*, 2012, pp. 721–729.