Knodle: Modular Weakly Supervised Learning with PyTorch

Anastasiia Sedova
University of Vienna
Vienna, Austria
anastasiia.sedova@univie.ac.at

Andreas Stephan
University of Vienna
Vienna, Austria
andreas.stephan@univie.ac.at

Marina Speranskaya
Ludwig Maximilian University of Munich
Munich, Germany
speranskaya@cis.lmu.de

Benjamin Roth
University of Vienna
Vienna, Austria
benjamin.roth@univie.ac.at

Abstract

Strategies for improving the training and prediction quality of weakly supervised machine learning models vary in how much they are tailored to a specific task or integrated with a specific model architecture. In this work, we introduce *Knodle*, a software framework that treats weak data annotations, deep learning models, and methods for improving weakly supervised training as separate, modular components. This modularization gives the training process access to fine-grained information such as data set characteristics, matches of heuristic rules, or elements of the deep learning model ultimately used for prediction. Hence, our framework can encompass a wide range of training methods for improving weak supervision, ranging from methods that only look at correlations of rules and output classes (independently of the machine learning model trained with the resulting labels), to those that harness the interplay of neural networks and weakly labeled data. We illustrate the benchmarking potential of the framework with a performance comparison of several reference implementations on a selection of datasets that are already available in *Knodle*.

1 Introduction

Most of today's machine learning success stories are built on top of huge labeled data sets. Creating and maintaining such data sources manually is a time-consuming, complicated and thus an expensive and error-prone process. Various research directions address the hunger for bigger and better datasets.

One of the most popular approaches that has recently gained traction is *weak supervision*. The learning algorithm is confronted with labels which are easy to obtain but are not guaranteed to be correct, and as such often demand denoising. Such weak labels are created, for example, with the use of regular expressions, keyword lists or external databases. Typically, methods for improving weakly supervised learning (and their respective implementations) are tailored towards domain-specific tasks or integrated with a specific model architecture. Examples include the attention-over-instances architecture introduced for relation extraction (Lin et al., 2016), an EM-based algorithm used for event extraction (Keith et al., 2017) or models of systematic label flips for named entity recognition (Hedderich et al., 2021). Such diversity and specificity of approaches makes it difficult to compare or transfer them across tasks without extensive adjustments dictated by the implementation, the task or the data set.

We introduce *Knodle*: a framework for *Kno*wledge-supervised *D*eep *Le*arning, i.e weak supervision with neural networks. The framework provides a simple tensor-driven abstraction based on PyTorch allowing researchers to efficiently develop methods for improving weakly supervised machine learning models and try them interchangeably to find the one that works the best for a given task. Within this work, we refer to a denoising method as any method that helps to improve weakly supervised learning regardless the type of noise or bias and the exact level of denoising (weak labels, weak rules etc).

The following points summarize *Knodle*'s main design goals:

- **Data abstraction.** A tensor-driven data abstraction subsumes a large number of input variants and is applicable to a diverse range of tasks.

- **Method independence.** A decoupled implementation of weak supervision denoising methods and prediction models enables comparability and accounts for domain-specific inductive biases.

- **Accessibility.** A high-level interface makes it easy to test existing methods, incorporate new ones and benchmark them against each other.

Several denoising algorithms are already included in *Knodle*. We also propose a new denoising algorithm, *WSCrossWeigh*, which extends *CrossWeigh* (Wang et al., 2019), a method for detecting mistakes in crowd-sourced annotation, to the weak supervision setting. The experiments demonstrate that it outperforms other existing methods on the majority of dataset s.

All implemented methods are tested on several datasets, also included in the *Knodle* ecosystem, and we discuss their performance. Each dataset exhibits different characteristics, such as the amount or the precision-recall balance of the used rules. Moreover, depending on the weakly labeled data set, methods for improving weak labels need to remove spurious matches in some cases, or generalize from them in others.

It is clear that such a diverse problem space should be paired with a rich pool of methods so that the most appropriate denoising method can be found for any task or dataset. *Knodle* allows to easily explore the spaces of weakly supervised learning settings and label improvement algorithms, and hopefully will facilitate a better understanding of the phenomena that are inherent to weakly supervised learning.

The framework is published as an open-source Python package `knodle` and available at `https://github.com/knodle/knodle`.

2 Related work

Many strategies have been introduced to reduce the need for large amounts of manually labeled data. Among these are *active learning* (Sun and Grishman, 2012), where automatically selected instances are manually annotated by experts, and *semi-supervised learning* (Agichtein and Gravano, 2000; Kozareva et al., 2008), where a small annotated dataset is combined with a large unlabeled one. Fine-tuning pretrained language models such as BERT (Devlin et al., 2018) shows good results if moderate to small amounts of annotations are available.

2.1 Weak supervision

In weak supervision, tedious expert work is replaced with easy to obtain, but potentially error-prone labels, that are usually derived from a set of heuristic rules. One of the most popular strategies of weakly supervised learning is *distant supervision*, which uses knowledge from existing data sources to annotate unlabeled data. The technique is used extensively for relation extraction (Craven and Kumlien, 1999; Mintz et al., 2009; Surdeanu et al., 2012; Riedel et al., 2013; Lin et al., 2016), where various knowledge databases, such as WordNet (Snow et al., 2004), Wikipedia (Wu and Weld, 2007) and Freebase (Mintz et al., 2009), are used as annotation sources.

When using heuristic rules, it is not uncommon that one sample turns out to be annotated by multiple rules. The most straightforward approach to resolve such cases is majority voting, which is used in early weak supervision algorithms (Thomas et al., 2011) as well as in more recent experiments (Krasakis et al., 2019; Boland and Krüger, 2019). However, majority voting does not deal with the different types of noise introduced by weak supervision, and more noise-specific algorithms are necessary. For example, the noise produced by *incomplete labels*, which stems from the incompleteness of weak supervision sources and often leads to an increased amount of false negatives, is commonly reduced by data manipulations, e.g. enhancing the knowledge base (Xu et al., 2013), a thorough construction of negative examples to balance the positive ones (Riedel et al., 2013), or explicitly modelling missing knowledge base information with latent variables (Ritter et al., 2013). The problem of *noisy features*, i.e. an increased amount of false positive labels stemming from overgeneralization, is often approached by using a relaxed distant supervision assumption (Riedel et al., 2010; Hoffmann et al., 2011), by active learning with additional manual expertise (Sterckx et al., 2014), with help of topic models (Yao et al., 2011; Roth and Klakow, 2013), as well as by using a combination of multiple methods (Roth, 2014).

Apart from that, methods treat the identified potentially noisy samples differently. They are either kept for further training with reduced weights (Jat et al., 2018; He et al., 2020), corrected (Shang, 2019) or eliminated (Qin et al., 2018). Thus, denoising methods vary significantly depending on the data and task, what makes the creation of a platform for comparison crucial.

2.2 Structure Learning

Structure learning assumes multiple weak labels per instance where each label is created by a so called *labeling function*. The goal is to learn a dependency structure within these labeling functions which motivates the term structure learning. Most labeling

functions are generated by human intuitions, motivating correlation and dependence between labeling functions. The first algorithm was implemented in the software package *Snorkel* (Ratner et al., 2017), which also implemented the data programming paradigm, allowing to programmatically create labeling functions. Subsequently improvements were made (Bach et al., 2017; Varma et al., 2019) and variations, such as semi-supdervised learning (Chatterjee et al., 2019; Maheshwari et al., 2020) were introduced.

2.3 Noise-aware learning

A common idea to mitigate single noisy labels is to build an architecture which accounts for noisy data. There are different approaches that model noise-robustness by adapting the loss function (Patrini et al., 2017). Examples include a generalization of cross-entropy and the mean absolute error (Zhang and Sabuncu, 2018) or the addition of a special noise layer to a neural network (Sukhbaatar et al., 2015). Many approaches are based on noise assumptions, such as on the assumption of symmetric label noise (van Rooyen et al., 2015). Another approach aims at finding and removing wrongly labeled samples from the training procedure. An example in this domain is given by the confidence learning framework *CleanLab*, which is based on the intuition that low-confidence predictions in cross-validation are more likely to be labeled wrongly (Northcutt et al., 2021). Note that most of these methods were built with the assumption that there is one label corresponding to each instance, while *Knodle* makes use of several weak signals per instance.

2.4 Crowdsourcing annotations

Another solution to reduce the cost of manual data supervision by experts is **crowdsourcing**. In order to increase the supervision accuracy for a task, most crowdsourcing experiments rely on annotations by multiple people, and the final label is defined by majority voting (Kosinski et al., 2012) or measuring the inter-annotator agreement (Tratz and Hovy, 2010). More sophisticated denoising strategies include anomaly detection (Eskin, 2000), annotator's reliability modelling (Dawid and Skene, 1979), Bayesian approaches (Raykar and Yu, 2012) and generative models (Hovy et al., 2013). Some mistakes can be identified by such methods. For example, mistakes consistently made by careful but biased people (Ipeirotis et al., 2010), or errors introduced by *spammers* (Raykar and Yu, 2012).

As both, automatically and human labeled data, are subject to noise and structural errors, many algorithms can be used for both domains. For example, the MACE algorithm (Hovy et al., 2013), initially proposed for improving noisy annotations from human annotators, was adapted to the setting of denoising automatically labeled data for named entity recognition (Rehbein and Ruppenhofer, 2017). With the same motivation, we introduce *WSCrossWeigh* (see Section 4 for more details). We demonstrate the usefulness of the *Knodle* framework to transfer algorithms for improving crowd-sourced annotations to weak supervision problems.

2.5 Frameworks

Knodle is based upon the ideas of several software frameworks. On a low level, *Knodle* is built on top of PyTorch (Paszke et al., 2017). As for design decisions, we followed several other high-level libraries that aim to ease the training and prediction experience. Namely, we drew inspiration from PyTorch lightning (Falcon, 2019), which in essence tries to remove the burdens of writing your own train loop, and Huggingface's Transformers library (Wolf et al., 2020), which gives easy access to various transformer-based architectures in a fixed manner, so that they can be effortlessly interchanged in code.

3 Weakly supervised learning with *Knodle*

The *Knodle* architecture provides a layer of abstraction that allows integrated label improvement and model training with weakly supervised learning signals in PyTorch. On the one hand, since *Knodle* has access to the information which rules matched for each sample, it is not restricted to methods that denoise only weak labels, such as Cleanlab (Northcutt et al., 2021). On the other hand, the *Knodle* abstraction also provides access to input and learned representations, and thus does not restrict denoising methods to rely on rule match correlations alone (as Snorkel (Ratner et al., 2017)). Moreover, access to the deep learning model enables the integration of denoising methods that use or manipulate the prediction model itself.

To the best of our knowledge, *Knodle* is the first framework to provide a modular architecture for interchangable application of a wide spectrum of denoising algorithms. For that reason we believe that it can become a testbed where different algorithms for improving the weakly supervised data are implemented and compared with each other to find the most fruitful task-to-denoising-method combination or to use it as a foundation for further studies.
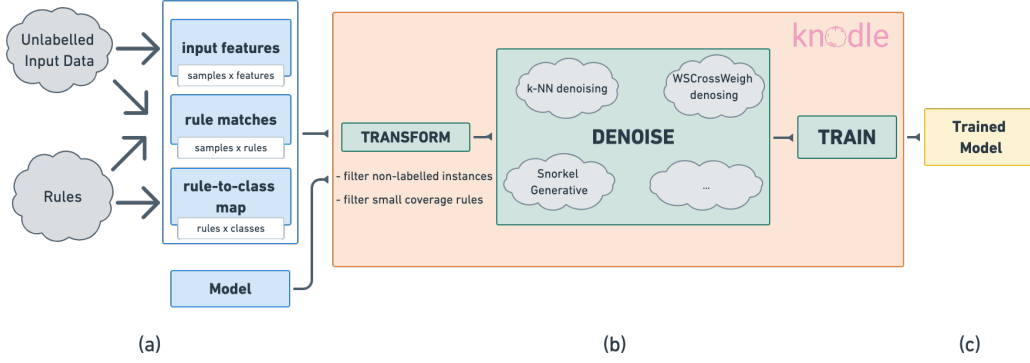
Figure 1: The figure gives an overview of our system. (a) represents the preprocessed input, given as tensors. (b) resembles the internals of *Knodle*. The `Trainer` classes introduced in Section 3.2 handle transformation, denoising and model training. Note that these three steps could be performed subsequently or subsumed in a single training step. Then, (c) shows the output, a trained PyTorch model.

The framework follows two main design principles, outlined below:

1. Tensor-based representations of input data and weak label matches

Similar to Pytorch models, where the data (input, labels) is already expected to be in tensor format, and the specific pre-processing that led to the tensor representation of the data is outside the scope of the deep learning model implementation, we choose to exclude the process of weak label generation from *Knodle*. Rather, we encode the information about weak labels in two tensors. One tensor contains information about which rules matched for each data instance, while another tensor describes the relationship between rules and output classes.

Formally, assume we have n samples, r rules and k classes. Rule matches are gathered in a binary matrix $Z \in \{0,1\}^{n \times r}$, where $Z_{ij} = 1$ if rule j matches sample i. The initial mapping from rules to the corresponding classes is given by another binary matrix $T \in \{0,1\}^{r \times k}$, $T_{jk} = 1$ if rule j is indicative of class k.

This separation between one tensor that contains rule matches and another tensor that translates them to labels allows *Knodle* to access this fine-grained information during training for certain denoising algorithms. This is in contrast to other approaches that treat weak supervision as learning from a noisy heuristic label matrix $Y_{heur} = ZT$ without direct access to the individual rules.

2. Separation of the prediction model from the weak supervision aspects.

Knodle requires a standard PyTorch model for a given prediction task. It is defined independent of the weak supervision aspects, such as rule types or denoising method. Therefore the same PyTorch model definition can be used for direct or weakly supervised training, and the two settings can easily be compared. However, even though the prediction model is defined separately, the denoising methods may have access to it during training. For example, cross-validation schemes such as WSCrossWeigh (see Section 4) can use the PyTorch model definition for data reweighting or label correction. This is in contrast to approaches that modularize denoising and training by first adjusting label confidences by using correlations between rules only and then training a model with the adjusted labels (Takamatsu et al., 2012; Ratner et al., 2017). Furthermore, *Knodle*'s design is much more flexible compared to approaches where denoising is so tightly integrated into the underlying prediction model architecture that it could not be changed (Sukhbaatar et al., 2015).

3.1 Handling of negative instances

Different tasks need a different logic to handle data samples where no rule matched. These samples are traditionally called *negative instances*. Whether unlabeled instances should be used for training (as an additional OTHER class) depends on the task at hand and should be configurable. For example, in knowledge base population (Surdeanu, 2013) there is only a small number of relevant target relations, and it is important to confidently identify sentences that do not contain any of the target relations (requiring negative instances as examples for the OTHER class). However, in spam classification with only two classes (spam and not spam) there are rules covering both possible outcomes, and there is no need for unlabeled instances and filtering them out is reasonable. Current weak supervision

frameworks provide only one of the two options: negative samples are either filtered out (Ratner et al., 2017) or included to the training dataset (Shu et al., 2020).

Knodle includes configurable functionality for handling such cases (allowing comparability of denoising methods across tasks with and without an OTHER class). From a technical point of view, there is a `filter_non_labeled` flag in a configuration file, which could be set to `False` if the negative instances should be filtered out. To make up for missing explicit annotations for negative samples, an additional `other_class` parameter is defined. Automatically all samples without a matching rule are set to belong to "other" class. Hence, the exact `other_class_id` could be either provided by the user or determined automatically by *Knodle*. These types of configurations are well encapsulated, allowing the specific model to deal with either input. The amount of negative instances that should included in the training set can be defined specifically for each denoising algorithm.

3.2 Implementation Details

Similar to the most popular deep learning frameworks, such as TensorFlow (Abadi et al., 2015) and PyTorch (Paszke et al., 2017), we realise learning as a mapping from input tensor(s) to output tensor(s) guided by a loss function that measures the quality of the learned mapping. However, while the most common solution is to represent the training data by a *design matrix* $X \in \mathbb{R}^{n \times d}$ (n instances represented by d feature dimensions) and a *label matrix* $Y \in \mathbb{R}^{n \times k}$ (k classes), input of *Knodle* are matrices X, Z and T described above. The heuristic labels themselves are calculated later during the weakly supervised learning using the information contained there. To ensure a seamless use, the weakly supervised algorithms need to be tightly integrated with automatic differentiation and optimization supported by PyTorch.

The denoising and training procedures are realised within `Trainer` classes. During initialization, they receive data, a possibly pre-initialized or pre-trained model, and a method-specific configuration, inheriting from `Config` containing information such as model training parameters, criterion, validation method, class weights, various options to handle cases where no rule matches discussed in 3.1 and others. The level of integration between denoising and training is different for each `Trainer`. Sometimes these procedures can be completely disentangled. For instance, the `SnorkelTrainer` firstly denoises

the input rules with Snorkel and, secondly, trains the classification model on the purified labels. Other methods highly integrate denoising and training with each other. An example is given by the `WSCrossWeighTrainer`, where several models are trained in oder to calculate sample weights as part of the denoising procedure before the final classifier is trained.

While in standard deep learning frameworks training can be executed by calling `model.train(X,Y)`, in *Knodle* the same functionality would be invoked with the following command (illustrates the Trainer with *k*-NN search, which we describe in Section 4):

```
kNNAggregationTrainer(model, X, Z,
        T, config).train()
```

The following code snippet shows an end-to-end process, starting from data loading, training and evaluation:

```
1   import torch
2   from knodle.trainer.knn_aggregation import \
3       kNNAggregationTrainer, kNNConfig
4
5   # load data in Knodle format
6   X_train, Z, T, X_test, Y_test = load_data()
7
8   # define custom config (or use default)
9   config = kNNConfig(epochs=2, k=3)
10
11  # initialize trainer
12  trainer = kNNAggregationTrainer(
13      model, X_train, Z, T, config
14  )
15
16  # train
17  trainer.train()
18
19  # evaluate
20  eval_dict = trainer.test(X_test, Y_test)
```

More detailed information about `kNNAggregationTrainer` as well as about other `Trainers` included to *Knodle* is provided in the next section.

4 Trainers

Knodle currently provides several out-of-the-box baselines and trainers, which we outline in the following section. All `Trainer` classes are compatible with any PyTorch model. As examples for PyTorch classifiers, *Knodle* provides code using logistic regression and HuggingFace's `transformers` (Wolf et al., 2020).

Majority Voting Baseline. As a simple baseline, the rules are directly applied to the test data without any additional model training. If several rules match,

the prediction is done based on the majority; ties are broken randomly. As was already mentioned in Section 2, it is one of the most basic approaches to denoise the data labeled by two or more rules or human annotators.

Trainer without Denoising. The simplest trained model is the `NoDenoisingTrainer`. The majority vote is computed on the training data and used to train the given model. This is the most direct use of the rule matches for training a classifier. To cover cases where several rules match, this trainer can be configured to either use a one-hot encoding of the winning label from the majority vote or a distribution over labels (relative to the number of matching rules).

Trainer with kNN Denoising. This `kNNAggregationTrainer` includes the label denoising method with a simple geometric interpretation. The intuition behind it is that similar samples should be activated by the same rules which is allowed by a smoothness assumption on the target space. The trainer looks at the k most similar samples sorted by, for example, TF-IDF features combined with L_2 distance, and activates the rules matching the neighbors to create a denoised $\hat{Z}$. Importantly, *Knodle* allows separate features for the model training and the neighborhood activation. This method also provides a way to activate rules for initially unmatched samples.

Trainer with Snorkel Denoising. *Knodle* provides a wrapper of the Snorkel system (Ratner et al., 2017) `SnorkelTrainer` which incorporates both generative and discriminative Snorkel steps. The generative step constitutes a denoising method in *Knodle*'s terminology, while the discriminative step corresponds to a prediction model. The structure within labels and rules, in our notation $P(Y,Z,T)$, is learned in an unsupervised fashion by the generative model. Afterwards, the final discriminative model, i.e. the prediction model, is trained with weak labels provided by the generative model, following the general *Knodle* design. Both steps are conveniently provided in a single method call.

Trainer with Weak Supervision CrossWeigh Denoising. Finally, we implemented our own algorithm for noise correction in weakly supervised data. It is based on the CrossWeight method (Wang et al., 2019) and included to *Knodle* as `WSCrossWeighTrainer`. While the original CrossWeigh method was proposed for mistakes identification in crowdworkers annotations, we extend it for denoising the weakly supervised data as well. In WSCrossWeigh we adopted the same logic for

estimating the reliability of weakly annotated data, but made some necessarily corrections specific to weakly supervised learning.

The main intuition behind WSCrossWeigh is the following: if a labeling rule corresponds to a wrong class and, therefore, annotates many samples in the training set with a wrong label, a machine learning model is likely to learn the incorrect pattern and to make similar mistakes when labeling the test samples. However, if we take a sufficiently big portion of data with samples *not* labeled by this rule, train the model on it, and then classify the samples matched by the rule, the predictions will contradict the initial *wrong* labels, and help us to trace the misclassified samples and reduce their importance in final classifier training.

As in the original CrossWeigh, the basic idea is similar to the *k*-fold cross-validation, where input data is split into *k* folds, each of which becomes, in turn, a test set, while the model is trained on the other folds. In WSCrossWeigh, however, the splitting is performed not randomly, but based on which rules match for the samples. Firstly, the rules are randomly split into K folds $\{r_1,...,r_k\}$ and, iteratively, each $fold_l$ is chosen to form a test set that is built from all samples matched this fold's rules. Other samples constitute a training set that is used for training the classification model. During the testing of the trained model on the hold-out fold samples, the predicted label $\hat{y}_i$ for each test sample x_i is compared to the label y_i originally assigned to x_i by weak supervision. If $\hat{y}_i \neq y_i$, this is taken as an indication that the sample x_i is likely to be potentially mislabeled, and its weights w_{x_i} is reduced by a value of an empirically estimated parameter ϵ. This procedure is repeated several times with different splits to detect misclassified samples more accurately.

The final classifier is trained on the whole reweighed training dataset. As a result, the more times the original y_i label of data sample x_i was suspected to be wrong, the smaller is its weight w_{x_i}, and, therefore, the smaller part it will play in the classifier training.

Along with other denoising algorithms, WSCrossWeigh was tested on the datasets described in Section 5 and showed quite promising results: it outperforms all other algorithms on three out of four datasets (for more details please see Section 6).

5 Datasets

Apart from denoising methods, *Knodle* includes a few datasets from previous works in the *Knodle*-specific tensor format in order to demonstrate the abilities of the framework. All datasets are rather simple, but have

dataset	classes	train / test samples	rules	avg. rule hits	class ratio
Spam	2	1586 / 250	10	1.63	0.47
Spouse	2	22254 / 2701	9	0.34	0.08
IMDb	2	40000 / 5000	6786	33.97	0.50
TAC-based RE	41	1937211 / 18660	182292	0.51	-

Table 1: Summary of data statistics. The average rule hits are computed on the train set. Class ratio describes the amount of positive samples in the test set for binary classification datasets, i.e. data skewedness.

their own peculiarities with respect to the respective Z and T matrices, that are worth investigating. The overview of dataset statistics is provided in Table 1.

Spam Dataset. The first task uses the YouTube comments dataset (Alberto et al., 2015). Here, the task is to classify whether a text is relevant to the video or holds spam, such as advertisement. The dataset has a small size of both train and test sets. Thus, a single wrongly labeled instance might have quite a big impact on the learning algorithm. We use the preprocessed version by the Snorkel team (Snorkel, 2020b). Among others, the rules were created based on keywords and regular expressions.

Spouse Dataset. This relation extraction dataset is based on the Signal Media One-Million News Articles Dataset (Corney et al., 2016). The task is to decide whether a sentence holds a spouse relation or not. Again, the preprocessed version by the Snorkel team is used (Snorkel, 2020a), so the results can be related to previous studies (Ratner et al., 2017). The rules are created via a set of known spouse relationships from DBPedia (Lehmann et al., 2014) as well as keywords and encoded language patterns. The difficulty of the Spouse dataset is its skewness: over 90% of samples in the test set hold a no-spouse relation.

IMDb Dataset. The third dataset is based on the well-known IMDb dataset (Maas et al., 2011), which consists of short movie reviews. The task is to determine whether a review holds a positive or negative sentiment. Despite the training set has labels, we do not use them in our experiments, but handle this data in an unsupervised fashion. To create the Z and T matrices, we use positive and negative keyword lists (Hu and Liu, 2004), with a total of 6800 keywords.

TAC-based Relation Extraction Dataset. Lastly, given the importance of distant supervision for relation extraction, we add a larger dataset with more relations (than just spouse). For development and test purposes the TACRED corpus annotated via crowdsourcing and human labeling from KBP (Zhang et al., 2017) is used. As human labels are not allowed

in weak training, the training is performed not on the TACRED dataset, but on a weakly-supervised noisy corpus built on TAC KBP corpora (Surdeanu, 2013; Roth, 2014), which was annotated with entity pairs extracted from Freebase (Google, 2014) with corresponding relations mapped to the 41 TAC relations. The amount of entity pairs per relation is limited to 10.000 and each entity pair is allowed to be mentioned in no more than 500 sentences. An important difference of this dataset to the other three is the presence of negative instances added to the dataset in equal proportion to the positive ones.

6   Experiments

The aim of *Knodle* is not to find the best denoising method in general. Rather, the goal is to find the method that improves weak labels most for a given task or dataset and its specific properties. Thus, *Knodle* supports experimentation to get a better understanding in which settings a certain method works well and when it does not.

6.1   Experimental Details

In all experiments, the DistilBert uncased model for English language (Sanh et al., 2019) provided by the HuggingFace [1] (Wolf et al., 2020) library is used as the prediction model. The optimization is performed with the AdamW optimizer (Loshchilov and Hutter, 2019) and a learning rate of $1e{-}4$. We employ a cross-entropy loss accepting a probability distribution over all labels as reference input whenever the output of a denoising algorithm is a distribution over weak labels (e.g. kNNAggregationTrainer, SnorkelTrainer). Reducing this representation to a single label (i.e. log-likelihood) would lead to a loss of weak signals, whereas a label distribution allows to exploit the information from Z and T to the fullest. Each model was trained for 2 epochs (unless stated otherwise), which was enough to receive a stable result.

---

[1]https://huggingface.co/

Mode	Spam Acc	Spouse P	R	F1	IMDb Acc	TAC-based RE P	R	F1
Majority vote	0.81	0.12	0.79	0.22	0.65	0.09	0.001	0.001
Majority + DistilBert	0.87	0.09	0.90	0.17	0.67	0.20	0.19	0.19
k-NN + DistilBert	**0.94**	0.12	0.86	0.21	0.50	0.10	0.11	0.10
Snorkel + DistilBert	0.88	0.13	0.70	**0.23**	0.50	-	-	-
WSCrossWeigh + DistilBert	**0.94**	0.09	0.69	0.16	**0.73**	0.25	0.27	**0.26**

Table 2: Results of the classifier training with different denoising methods on the test sets of datasets included in *Knodle*. [†]The neighbors were searched with Approximate Nearest Neighbors (Bernhardsson, 2015) because of computation complexity of k-NN search.

For the k-NN algorithm, nearest neighbors were found using the cosine similarity of TF-IDF features based on a dictionary of 3000 words, and the number of k neighbors is treated as a hyper-parameter. In our experiments, we used $k = 2$ except where otherwise noted. Hyperparameters for the WSCrossWeigh denoising algorithm are the number of folds the data is be split into, the number of partitions (that is, how many times the splitting for mistake estimation is done) and a weight-reducing rate (the value, by which the initial sample weights are reduced to each time the sample is predicted wrongly). These parameters are tuned for each dataset individually. The following best parameter values were found empirically: ($folds = 3$, $partitions = 10$ and $\epsilon = 0.3$) for the Spam dataset, (3, 2 and 0.3) for the Spouse dataset and (2, 25, 0.7) for the IMDb dataset. Apart from that, *Knodle* provides the opportunity to train the cross-validated sample weights with a model different from the final classifier. In our experiments, the weights were calculated using a Bidirectional LSTM with GloVe Embeddings (Pennington et al., 2014), while the final training was performed with DistilBert using the same settings as in the experiments with other denoising methods. The only difference is the number of epochs on the TAC-based dataset: the best results were obtained with 1 DistilBert epoch.

6.2 Results

An overview of the results is given in Table 2. In the Spam dataset, all denoising methods show an improvement over the simple majority vote baseline. The data-adaptive k-NN and WSCrossWeigh methods perform best in this setting. Snorkel and standard majority voting followed by DistilBert fine-tuning overfit to the noisy majority votes. This becomes obvious with the observation that Snorkel achieves a score of 0.93

with a simple logistic regression discriminative model. Interestingly, k-NN performs well which can serve as a proof for the reliability of neighboring labels.

Compared to the Spam dataset, the Spouse dataset is much larger. As the task is to find sentences holding spouse relations, we relate all metrics to the *is-spouse* relation. Note that the *non-spouse* relation remains in this case completely disregarded. Furthermore, the class ratio equals 0.08 shows that *is-spouse* is the complicated class of interest. On average, 0.34 rules hit per instance, meaning that almost 70% of the data match no rule. In these cases, majority vote uses a random vote which oversamples the *is-spouse* relation, rendering a high recall but low precision. We found that the rule matches overrepresent the *is-spouse* class as they are closer to a class ratio of 0.5 than to the true class ratio of 0.9. Thus, the additional model training magnifies overfitting towards the *is-spouse* class which, again, is expressed by increased recall and lower precision. The only denoising system that generalizes is Snorkel. One possible explanation could be that it is the only method that provides explicit rule denoising.

For IMDb, the majority vote shows that the rules have rather low quality on their own, but an additional trained model on top manages to generalize beyond the given labels. In contrast, denoising with the k-NN algorithm only aggravates the problems inherent to labels as the classifier's performance drops down to a random vote (50% accuracy). This behaviour can be explained by the high density of rule hits: on average, no less than 33 keywords match for each sentence, which means that already for $k = 1$ many neighbors are added and that the propagation of imprecise labelings overrules the expected benefits of k-NN. In general, there are cues that k-NN might useful in cases where the weak labels are already rather reliable but fail in cases where weak labels are too noisy. The

Snorkel based denoising does not perform well on IMDb dataset as well, which can be explained by the lack of dependencies between the rules that the Snorkel system relies on. However, WSCrossWeigh appears to be very robust to these data characteristics, the large amount of rules seems to help tease out and mutually reinforce the data characteristics associated with a specific label in cross-validation.

The distantly supervised TAC-based RE dataset turns out to be the most complicated dataset among all because of a larger size of samples n and a larger number of rules r. Due to its specificity, there are almost no rule matches (entity pairs from the seed KB) on the test set, implying that the simple majority baseline has scores close to 0. Training with DistilBert improves the result, however the performance remains considerably worse than for the data sets discussed above. On the contrary the WSCrossWeigh method that not directly denoise the rules, but downweigh the mislabeled data samples is still able to improve the results. Snorkel denoising could not be performed on this dataset on a machine with CPU frequency of 2.2GHz with 40 cores due to the immense amount of rules without the data manipulations we want to avoid (such as significantly reducing the number of rules). The computation of distances between almost 2 millions instances, which are necessary to determine the nearest neighbors, also turned out to be extremely memory- and time-consuming, explaining why k-NN algorithm was also not performed. Instead, we work around this by applying an *approximated k*-NN algorithm. In our experiments we used the Annoy library (Bernhardsson, 2015) and $k = 3$ parameter. The poor performance of approximated k-NN could be explained by a small average of rule hits in the TAC-based RE data set; the possible approximation losses are also not to be neglected. In contrast, the WSCrossWeigh method performs quite well. Our explanation is that WSCrossWeigh does not directly denoise the rules, but down-weighs samples it is less confident about. This makes this approach more robust in cases where the rules are very noisy.

7   Conclusion

This work introduces the Knowledge-supervised Deep Learning framework *Knodle*. *Knodle* provides a unified interface to work with multiple weak labeling sources, so that they can be seamlessly integrated with the training of deep neural networks. This is achieved by a tensor-based input format and a intuitive separation of weak supervision aspects and model

training. The framework facilitates experimentation that helps researchers to gain better insights into the correspondence between characteristics of weak supervision problems, and the effectiveness of methods for improving weakly supervised learning. From a practical perspective, *Knodle* can be used to compare different denoising methods and select the one that gives the best result for a specific task.

Knodle's modular approach makes it easy to add new data sets and denoising algorithms. Adding functionality to *Knodle* is straightforward, and we do hope that it will encourage researchers to create their own algorithms to improve learning with weakly annotated data, and incorporate them into the *Knodle* framework.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, DL '00, page 85–94, New York, NY, USA. Association for Computing Machinery.

T C Alberto, J V Lochter, and T A Almeida. 2015. TubeSpam: Comment Spam Filtering on YouTube. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, pages 138–143.

Stephen H Bach, Bryan Dawei He, Alexander Ratner, and Christopher Ré. 2017. Learning the Structure of Generative Models without Labeled Data. *CoRR*, abs/1703.0.

E. Bernhardsson. 2015. Annoy on github. Last accessed 23 April 2021.

K. Boland and F. Krüger. 2019. Distant supervision for silver label generation of software mentions in social scientific publications. In *BIRNDL@SIGIR*.

Oishik Chatterjee, Ganesh Ramakrishnan, and Sunita Sarawagi. 2019. Data Programming using Continuous and Quality-Guided Labeling Functions. *CoRR*, abs/1911.0.

D. Corney, M. Albakour, Miguel Martinez-Alvarez, and Samir Moussa. 2016. What do a million news articles look like? In *NewsIR@ECIR*.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, page 77–86. AAAI Press.

A. P. Dawid and A. M. Skene. 1979. Maximum likelihood estimation of observer error-rates using the em algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):20–28.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Eleazar Eskin. 2000. Detecting errors within a corpus using anomaly detection. In *1st Meeting of the North American Chapter of the Association for Computational Linguistics*.

WA Falcon. 2019. Pytorch lightning. *GitHub. Note: https://github.com/PyTorchLightning/pytorch-lightning Cited by*, 3.

Google. 2014. Freebase data dumps. https://developers.google.com/freebase/data.

Zhengqiu He, Wenliang Chen, Yuyi Wang, Wei Zhang, Guanchun Wang, and Min Zhang. 2020. Improving neural relation extraction with positive and unlabeled learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:7927–7934.

Michael A. Hedderich, Dawei Zhu, and Dietrich Klakow. 2021. Analysing the noise model error for realistic noisy label data.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 541–550, Portland, Oregon, USA. Association for Computational Linguistics.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130, Atlanta, Georgia. Association for Computational Linguistics.

Minqing Hu and Bing Liu. 2004. Mining and Summarizing Customer Reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA. Association for Computing Machinery.

Panos Ipeirotis, Foster Provost, and Jing Wang. 2010. Quality management on amazon mechanical turk. *In:Proceedings of the ACM SIGKDD Workshop on Human Computation*.

Sharmistha Jat, Siddhesh Khandelwal, and Partha Talukdar. 2018. Improving distantly supervised relation extraction using word and entity based attention.

Katherine A. Keith, Abram Handler, Michael Pinkham, Cara Magliozzi, Joshua McDuffie, and Brendan O'Connor. 2017. Identifying civilians killed by police with distantly supervised entity-event extraction. *CoRR*, abs/1707.07086.

Michal Kosinski, Yoram Bachrach, Gjergji Kasneci, Jurgen Van-Gael, and Thore Graepel. 2012. Crowd iq: Measuring the intelligence of crowdsourcing platforms. *Proceedings of the 3rd Annual ACM Web Science Conference, WebSci'12*.

Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. 2008. Semantic class learning from the web with hyponym pattern linkage graphs. In *Proceedings of ACL-08: HLT*, pages 1048–1056, Columbus, Ohio. Association for Computational Linguistics.

Antonios Minas Krasakis, E. Kanoulas, and G. Tsatsaronis. 2019. Semi-supervised ensemble learning with weak supervision for biomedical relationship extraction. In *AKBC*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, and Christian Bizer. 2014. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6.

Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, and Maosong Sun. 2016. Neural relation extraction with selective attention over instances. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2124–2133, Berlin, Germany. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association*

for Computational Linguistics: Human Language Technologies, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Ayush Maheshwari, Oishik Chatterjee, KrishnaTeja Killamsetty, Rishabh Iyer, and Ganesh Ramakrishnan. 2020. Data Programming using Semi-Supervision and Subset Selection.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011, Suntec, Singapore. Association for Computational Linguistics.

Curtis G Northcutt, Lu Jiang, and Isaac L Chuang. 2021. Confident Learning: Estimating Uncertainty in Dataset Labels.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. *NIPS Workshop*.

Giorgio Patrini, Alessandro Rozza, Aditya Menon, Richard Nock, and Lizhen Qu. 2017. Making Deep Neural Networks Robust to Label Noise: a Loss Correction Approach.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Pengda Qin, Weiran Xu, and William Yang Wang. 2018. Robust distant supervision relation extraction via deep reinforcement learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2137–2147, Melbourne, Australia. Association for Computational Linguistics.

Alexander Ratner, Stephen H Bach, Henry R Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid Training Data Creation with Weak Supervision. *CoRR*, abs/1711.1.

Vikas C. Raykar and Shipeng Yu. 2012. Eliminating spammers and ranking annotators for crowdsourced labeling tasks. *Journal of Machine Learning Research*, 13(16):491–518.

Ines Rehbein and Josef Ruppenhofer. 2017. Detecting annotation noise in automatically labelled data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1160–1170, Vancouver, Canada. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part III*, ECML PKDD'10, page 148–163, Berlin, Heidelberg. Springer-Verlag.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia. Association for Computational Linguistics.

Alan Ritter, Luke Zettlemoyer, Mausam, and Oren Etzioni. 2013. Modeling missing data in distant supervision for information extraction. *Transactions of the Association for Computational Linguistics*, 1:367–378.

Brendan van Rooyen, Aditya Krishna Menon, and Robert C Williamson. 2015. Learning with Symmetric Label Noise: The Importance of Being Unhinged.

Benjamin Roth. 2014. *Effective distant supervision for end-to-end knowledge base population systems*. Ph.D. thesis, Saarland University.

Benjamin Roth and Dietrich Klakow. 2013. Feature-based models for improving the quality of noisy training data for relation extraction. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, CIKM '13, page 1181–1184, New York, NY, USA. Association for Computing Machinery.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

Yuming Shang. 2019. Are noisy sentences useless for distant supervised relation extraction?

Kai Shu, Subhabrata Mukherjee, Guoqing Zheng, Ahmed Hassan, Milad Shokouhi, and Susan Dumais. 2020. Learning with weak supervision for email intent detection. pages 1051–1060.

Snorkel. 2020a. Detecting spouse mentions in sentences. Last accessed 25 February 2021.

Snorkel. 2020b. Snorkel intro tutorial: Data labeling. Last accessed 25 February 2021.

Rion Snow, Daniel Jurafsky, and Andrew Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*, volume 17.

Lucas Sterckx, Thomas Demeester, Johannes Deleu, and Chris Develder. 2014. Using active learning and semantic clustering for noise reduction in distant supervision. In *NIPS 2014*.

Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. 2015. Training Convolutional Networks with Noisy Labels.

Ang Sun and Ralph Grishman. 2012. Active learning for relation type extension with local and global data views. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, page 1105–1112, New York, NY, USA. Association for Computing Machinery.

M. Surdeanu. 2013. Overview of the tac2013 knowledge base population evaluation: English slot filling and temporal slot filling. *Theory and Applications of Categories*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 455–465.

Shingo Takamatsu, Issei Sato, and Hiroshi Nakagawa. 2012. Reducing wrong labels in distant supervision for relation extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 721–729, Jeju Island, Korea. Association for Computational Linguistics.

Philippe Thomas, Illés Solt, Roman Klinger, and Ulf Leser. 2011. Learning protein–protein interaction extraction using distant supervision. In *Proceedings of Workshop on Robust Unsupervised and Semisupervised Methods in Natural Language Processing*, pages 25–32, Hissar, Bulgaria. Association for Computational Linguistics.

Stephen Tratz and Eduard Hovy. 2010. A taxonomy, dataset, and classifier for automatic noun compound interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 678–687.

Paroma Varma, Frederic Sala, Ann He, Alexander Ratner, and Christopher Ré. 2019. Learning Dependency Structures for Weak Supervision Models.

Zihan Wang, Jingbo Shang, Liyuan Liu, Lihao Lu, Jiacheng Liu, and Jiawei Han. 2019. CrossWeigh: Training named entity tagger from imperfect annotations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5154–5163, Hong Kong, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M Rush. 2020. HuggingFace's Transformers: State-of-the-art Natural Language Processing.

Fei Wu and Daniel S. Weld. 2007. Autonomously semantifying wikipedia. In *CIKM*, page 41– 50. ACM.

Wei Xu, Raphael Hoffmann, Le Zhao, and Ralph Grishman. 2013. Filling knowledge base gaps for distant supervision of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 665–670, Sofia, Bulgaria. Association for Computational Linguistics.

Limin Yao, Aria Haghighi, Sebastian Riedel, and Andrew McCallum. 2011. Structured relation discovery using generative models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1456–1466, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP 2017)*, pages 35–45.

Zhilu Zhang and Mert R Sabuncu. 2018. Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels.