SepLL: Separating Latent Class Labels from Weak Supervision Noise

Anonymous ACL submission

Abstract

Weak Supervision is a common approach that aims to tackle the need for large labeled datasets. In this setting, *labeling functions* automatically assign heuristic, often noisy, labels to data samples. In this work, we provide a method for learning from weak labels by separating two types of mutually exclusive information associated with the labeling functions: information related to the target label and information specific to one labeling function only. Both types of information are reflected to different degrees by all labeled instances. In contrast to previous works that aimed at correcting or removing wrongly labeled instances, we learn a branched deep model that uses all data as is, but splits the labeling function information in the latent space. Specifically, we propose the end-to-end model *SepLL* which extends a transformer classifier by introducing a latent space for labeling function specific and task-specific information. The learning signal is only given by the labeling functions matches, no pre-processing or label model is required for our method. Notably, the task prediction is made from the latent layer without any direct task signal. Experiments on the Wrench text classification tasks show that our model is competitive with the state-of-the-art, and yields a new best average performance.

1 Introduction

In recent years, large language modelling approaches have proven their applicability to a wide range of tasks, mainly due to the pre-training and fine-tuning paradigm. This has created a need for large labeled datasets as training on these datasets enables models to achieve state-of-the-art performance. However, obtaining manually created labels is expensive, tedious and often requires expert knowledge. As a consequence, significant areas of research are devoted to addressing this challenge by minimizing the need for labeled data. For example, research directions include transfer learning (Ruder et al., 2019) or few-shot learning (Brown et al., 2020). Another research direction to address this challenge is weakly supervised learning. The idea is to use human intuitions, heuristics and existing resources, e.g., related databases, to create weak (noisy) labels.

Several approaches have been proposed to increase the quality of the resulting labels. For example, Ratner et al. (2017) use generative modeling to learn a probability distribution over the labeling function matches, i.e., weak labels, and unknown true labels in order to denoise the labels and subsequently train a classifier. Recently, several works use student-teacher schemes that use knowledge inherent to pre-trained models (Karamanolakis et al., 2021; Cachay et al., 2021; Ren et al., 2020). Usually a summary statistic of weak labels, such as majority vote, is used as ground truth and iteratively updated during training, for example by employing a regularization based on the prediction confidence of the model (Yu et al., 2021). Thus, most methods share the property that the weak labels, i.e., the learning signals, are transformed or updated throughout the learning process.

Instead of updating the weak labeling, we want to keep the weak labels as-is and make use of a different intuition. Each labeling function provides information relevant to the prediction task but also information only related to the function itself. The idea is to view these two types of information as mutually exclusive and build a model which separates them.

To this end, we propose *SepLL*, an end-to-end model which stacks two branched latent layers, representing target-task-related and labeling-function-related information, on top of a transformer encoder and recombines them for predicting labeling function occurrences. Then, the learning signal is only given by the weak labels as is. Notably, the task prediction is performed from the latent space without any direct supervision. Multiple informa-

tion routing strategies are employed to improve the separation.

In order to evaluate the performance, experiments on the text classification tasks of the WRENCH benchmark (Zhang et al., 2021) are performed. The standard model achieves state-of-the-art performance when compared to standalone models as well as when combined and compared with the self-improvement method Cosine (Yu et al., 2021). An ablation study shows the importance of each information routing strategy. The experiments show that in addition to its task performance, the model is able to memorize the labeling function information.

The contributions can be summarized in three parts: 1) We introduce a new intuition what type of information labeling functions provide and turn it into a method, *SepLL*, reflecting the intuition in the latent space. 2) We provide an analysis through experiments on the WRENCH benchmark, an ablation study and an in depth analysis of the two latent spaces. 3) We provide the code and a suitably transformed version of the input data. [1]

2 Related Work

Weak Supervision. A main concern in machine learning is that a large amount of labeled data is needed in order to train models that achieve a state-of-the-art performance. Among others, the field of weak supervision aims to address this issue. The idea is to formalize human knowledge or intuitions into weak supervision sources, called labeling functions, which can be used to produce weak labels. Examples of labeling functions are heuristic rules, e.g., keywords, regular expressions, other pre-trained classifiers or knowledge bases in distant supervision (Mintz et al., 2009; Hoffmann et al., 2011; Takamatsu et al., 2012).

A main challenge that appears in a weak supervision setting is how to create accurate labeling functions and how to unify and denoise them. Majority vote, Snorkel (Ratner et al., 2017) (based on data programming) and Flying Squid (Fu et al., 2020) are methods that compute weak labels based on generative models over the labeling function matches and unknown true labels. These models are referred to as label models. Subsequently so called end-models, e.g., BERT-style classifiers (Devlin et al., 2019), or methods dedicated to noisy training labels are used to train a final model.

[1] To be published upon acceptance.

Recently, neural methods, including the use of pre-trained models, gained more traction. (Cachay et al., 2021) use a classifier and a probabilistic encoder for the labeling function matches and optimize them using a noise-aware loss. Similarly, Ren et al. (2020) combine a classifier and a attention-based denoiser, but also include unlabeled samples. Yu et al. (2021) introduced Cosine, which is a method to self-optimize classification models. They leverage contrastive learning and confidence regularization, i.e., high-confidence samples, to optimize a model's performance.

Other approaches use additional signals. For instance, ImplyLoss (Awasthi et al., 2020) uses access to exemplars, i.e., single, correctly labeled samples and ASTRA (Karamanolakis et al., 2021) follows an attention based student-teacher mechanism with an additional supervision of a few manually annotated labeled samples. Zhu et al. (2022) uses a meta self-refinement approach which makes use of access to the validation performance.

Our experiments are built on the Weak Supervision Benchmark (Wrench) (Zhang et al., 2021), which is a framework that aims to provide a unified and standardized way to run and evaluate weak supervision approaches. A wide range of tasks, datasets and implementations of weak supervision methods are available.

Latent Variable Modelling. There is work regarding latent variable modelling in other areas of machine learning, which has influenced the rationale behind this work. Research in representation learning has focused on modelling mutually independent factors of variation, e.g., color in computer vision, explicitly in some latent space. Often this is called *disentanglement* (Bengio et al., 2013). This is transferable to our setting as we aim to obtain the task prediction as a disentangled factor. An important early technique is Independent Component Analysis (ICA) (Comon, 1994). Kingma and Welling (2014) introduced variational autoencoders (VAE's) to neural networks, allowing complex data distributions to be represented as simple distributions in the latent space. An extension is given by β-VAE (Higgins et al., 2017), which is more suitable for disentanglement. In addition, there has been progress on theoretical work, which aims to give an insight on what information is identifiable by using self-supervised learning (SSL), e.g., Zimmermann et al. (2021) prove under certain assumptions that it inverts the data generation process. An

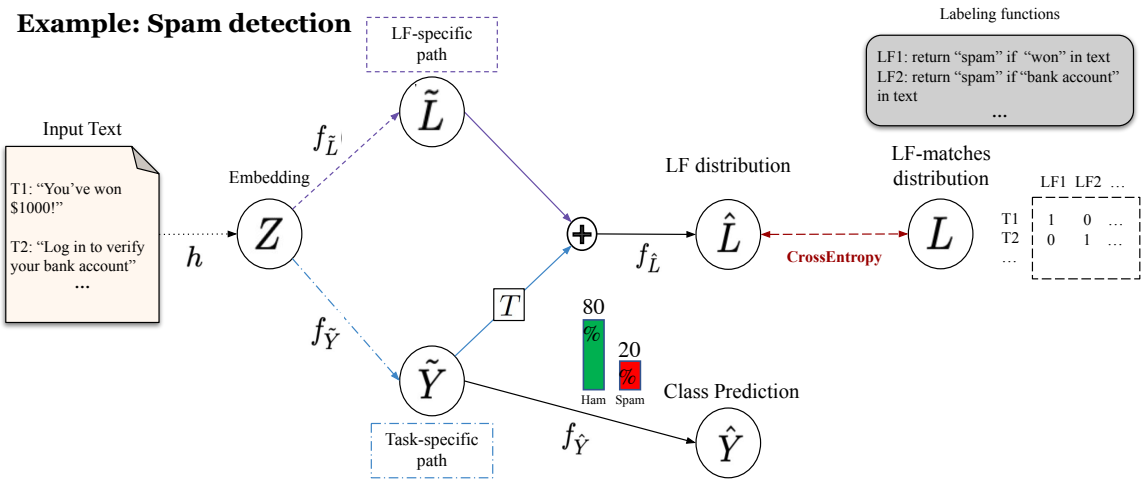

Figure 1: Overview of *SepLL*. Text gets embedded into Z by a Transformer encoder, and then this representation is split into labeling function-specific and task-specific information. The task-specific information is translated back into the LF space and re-combined into $\hat{L}$. A cross-entropy loss between the distribution of labeling function matches L and $\hat{L}$ is minimized. The latent task prediction $\hat{Y}$ can be used for classification.

interesting perspective is the separation of content and style, e.g., the animal in a picture (content) and the camera angle of the image (style). Under milder assumptions as in Zimmermann et al. (2021), it is proved by von Kügelgen et al. (2021) that this separation is achieved using SSL. Mentioned works are not directly applicable to our task, because we want to separate general aspects of labeling functions, which are useful for prediction tasks, from labeling function specific aspects.

3 Method

The motivation of this work is that each labeling function provides two types of information.On the one hand, it provides information about the target task, e.g., spam detection, and on the other hand it provides information related to the labeling function itself. This translates to our model, called *SepLL*, which aims to separate these two types of signals in a latent space. Figure 1 provides an overview of *SepLL*.

In this section, we first introduce some notation and then describe the architecture of *SepLL*. Following, the training mechanisms, which aim to support the separation of the two information types are discussed.

3.1 Problem Setup and Notation

In general, the goal is to solve classification tasks, e.g., spam detection asks whether a text is spam or not. The input space is denoted by X and the unknown labels are denoted by $Y = \{y_1, \ldots, y_c\}$. Additionally m labeling functions $l_i : X \to \{y\} \cup \emptyset, i = 1, \ldots, m$ are given where each labeling function (LF) either assigns a dedicated specific label $y \in Y$ to a sample or abstains from labeling. If a label is assigned, we say a labeling function *matches* a sample. The task is to use input X and labeling functions l_i to learn a mapping $X \to Y$. We use the format of the Knodle (Sedova et al., 2021) framework, where each labeling function is encoded as a labeler for exactly one class. This is in contrast to other conventions where a single labeling function is allowed to label multiple classes, e.g., Ratner et al. (2016). This convention can easily be transformed into our setting, by splitting multi-class LFs into multiple class-specific LFs. The matching matrix $L \in \{0,1\}^{n \times m}$ describes whether labeling function j matches sample i by setting $L_{ij} = 1$, otherwise $L_{ij} = 0$. The mapping matrix $T \in \{0,1\}^{m \times |Y|}$ reflects a simple mapping between labeling function i and class j by $T_{ij} = 1$, otherwise $T_{ij} = 0$.

3.2 Basic Model

First, the model transforms input text X into a latent representation Z using an encoder $h : X \to \mathbb{R}^d$. In this case, a pre-trained transformer encoder transforms input text $x \in X$ into the <CLS>-token embedding $z = h(x) \in \mathbb{R}^d$.

Following, there are two transformations $f_{\tilde{Y}} : \mathbb{R}^d \to \mathbb{R}^{|Y|}$ and $f_{\tilde{L}} : \mathbb{R}^d \to \mathbb{R}^m$, which are realized by two multi-layer perceptrons. The goal is

to train the model such that $f_{\tilde{Y}}(x)$ reflects the task information and $f_{\tilde{L}}(x)$ the remaining LF-related information. Note that the resulting output represents the log-space and the transformation to probabilities happens later.

Afterwards, the two latent layers are combined again and compared to the training signal, which is purely given by the LF matches L. The T matrix is used to map the target label information to the corresponding LF information by

$$f_{\hat{L}}(z) = f_{\tilde{Y}}(z)T^{\top} + f_{\tilde{L}}(z) \in \mathbb{R}^m \qquad (1)$$

where $z = h(x)$ is the latent representation. Crucially, the T matrix establishes the connection between task path and LF path.

In order to run the optimization we compare the combined signals to the labeling functions matches. Therefore we define the *LF distribution* as the normalized L matrix, i.e., $P_{ij} = \frac{L_{ij}}{\sum_k L_{ik}}$.

Finally, the loss is computed as the cross entropy between the labeling function distribution and the prediction

$$\mathrm{CE}(P, Q) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{l} p_{ij}\log(q_{ij}) \qquad (2)$$

where the prediction probability Q is given by a softmax activation over $f_{\hat{L}}$.

The task prediction is computed using a softmax activation on the latent task signal, i.e.,

$$P(y_i|x) = \left(f_{\hat{Y}}(z)\right)_i = \frac{e^{\left(f_{\tilde{Y}}(z)\right)_i}}{\sum_{j=1}^{c} e^{\left(f_{\tilde{Y}}(z)\right)_j}} \qquad (3)$$

where $z = h(x)$ is again the latent representation. Thus, no direct supervision is performed.

3.3 Latent information routing

Clearly, the separation could easily collapse by just using the labeling function path, i.e., there is no apparent reason why the LF prediction $\hat{L}$ is not only based on the LF-specific path. Therefore we introduce three schemes supporting the separation of information. Firstly a regularization, secondly an adaption of the learning label and thirdly the inclusion of unlabeled samples are discussed.

Regularization. The easiest solution to the problem is to introduce standard regularization schemes. We consider the L_2 regularization as suitable because it encourages the optimization to put similar weight on the parameters. We test two types of L_2 regularization. Firstly, standard weight decay is used which employs an L_2 regularization on all parameters, including the transformer encoder. Secondly, an additional L_2 regularization is applied to $f_{\tilde{L}}$. The goal is to regularize the labeling function path such that more weight is put on the class prediction path. In the experimental part we refer to the two types as weight decay and L_2 regularization, respectively.

Noise Injection. Our hypothesis is that the optimization routine puts weight on the task-specific path if two labeling functions belonging to the same class match simultaneously. But most of the time samples are only matched by a single labeling function. Thus, we inject noise in the form of additional *"hallucinated"* matches into the labeling function matrix L. If a sample is matched by a labeling function, we create a match for all other labeling functions for the same class with a probability proportional to a random factor $\lambda \in [0, 1]$, which is a hyperparameter.

Usage of Unlabeled Samples. It has been previously shown, e.g., by Ren et al. (2020) and Yu et al. (2021) that it is effective in weak supervision settings to make use of unlabeled samples X_U, i.e., samples where no labeling function matches. Apart from a performance increase it was shown that the learning gets more robust. Thus, we adapt this semi-supervised learning approach. In order to comply with our basic model, we need to create a labeling function distribution Q_U. We take the simplest idea and define Q_U as the uniform distribution over the labeling function matches, i.e., $(Q_U)_{ij} = \frac{1}{m}$ for all unlabeled samples i and labeling functions j.

4 Experimental Setup

Before analysing the experiments, we discuss the experimental setup. More specifically, an overview of the used datasets, baselines, hyperparameters and some notes on reproducibility and implementational details are given.

4.1 Datasets

For the experiments we used all text classification datasets that are currently included in the Wrench benchmark. As shown in Table 1 the datasets reflect varying properties, such as sample size, coverage or the amount of labeling functions.

Five out of eight datasets represent binary classi-

Dataset	#Classes	#LF's	# Train	# Coverage	# Dev	# Test
IMDb	2	9	20000	88%	2500	2500
Yelp	2	15	30400	83%	3800	3800
Youtube	2	10	1586	88%	120	250
SMS	2	73	4571	41%	500	500
AGNews	4	9	96000	69%	12000	12000
TREC	6	68	4965	95%	500	500
Spouse	2	9	22254	26%	2811	2701
SemEval	9	164	1749	100%	178	600

Table 1: Statistics describing the datasets as they are used in the WRENCH framework. Coverage is computed on the train set by dividing the number of samples having at least one match by the number of samples.

fication problems. These are: (i) **Youtube (Alberto et al., 2015)**, which consists of text comments from YouTube videos, each labeled as spam or non-spam, (ii) **SMS (Almeida et al., 2011)**, which is a mobile phone spam corpus, which contains real SMS messages that are spam or non-spam, (iii) **IMDb review dataset (Maas et al., 2011)** contains reviews from IMDb and each review is labeled as positive or negative, (iv) **Yelp (Zhang et al., 2015)** consists of positive or negative reviews from the Yelp Dataset Challenge 2015, (v) **Spouse (Corney et al., 2016)** , which is a relation classification dataset, where we decide for each sentence if it contains a spouse relation or not. Three datasets correspond to multi-label problems: (i) **AGNews (Zhang et al., 2015)** consists of news articles classified into 4 classes, (ii) **TREC (Li and Roth, 2002)** is a question classification dataset, where the questions are classified into 6 labels, and (iii) **SemEval (Hendrickx et al., 2010)** contains sentences collected from the Web and the task is to identify the relation between two nominals tagged in each sentence among 9 types of semantic relations.

4.2 Baselines

We compare *SepLL* to several traditional and state-of-the-art models that can be categorized in 4 approach types: supervised, statistical, neural and cosine based (see Table 2). Wherever possible the RoBERTa-base (Liu et al., 2020) backbone model is used.

For the supervised approach (**Gold + RoBERTa**), which serves as an upper bound, we perform a standard fine-tuning of a RoBERTa (Liu et al., 2020) pre-trained model using gold labels.

The statistical approaches are: (i) **Majority Vote (MV)** As the name suggests, majority vote picks the label indicated by the majority of labeling function matches. In case there is no match, a random label is chosen, (ii) **Data Programming (DP)** employs Snorkel DP (Ratner et al., 2017) to obtain weak labels and (iii) **Flying Squid (FS)** (Fu et al., 2020) is a label model making use of so-called triplet methods.

In the neural category of baselines there are three dedicated end-to-end models: (i) **WeaSEL** (Cachay et al., 2021) uses a classifier and a probabilistic encoder combined with a noise-aware loss function, (ii) **Denoise** (Ren et al., 2020) uses a classifier and an attention-based denoiser, mutually optimizing each other, (iii) **KnowMAN** is an adversarial architecture that aims to learn representations that are invariant to the labeling function signals (März et al., 2021). In addition to learning a classifier for the end task, a labeling function discriminator is trained and its negative gradient is used to update a shared feature extractor.

We also employ models that combine the labels obtained by MV, DP and FS with the RoBERTa model (**MV+RoBERTa**, **DP+RoBERTa** and **FS+RoBERTa** respectively) (Liu et al., 2020).

Cosine (Yu et al., 2021) takes a pre-trained classifier and uses contrastive learning and confidence regularization to improve the performance of the classifier. Cosine is a model-agnostic method for self-training that can be combined with any classifier and is not specific to weak supervision. It has been observed that Cosine particularly helps with standard weak supervision methods like majority voting and data programming, and we include the best-performing combinations from Wrench (MV+Cosine, DP+Cosine), as well as *SepLL*+Cosine in our experiments. Additional information on the baselines is given in appendix A.

	IMDb	Yelp	Youtube	SMS	AGNews	Trec	Spouse	Semeval	Avg.
Supervised									
Gold+RoBERT†	93.25	97.13	95.68	96.31	91.39	96.68	-	93.23	88.58
Statistical									
MV†	71.04	70.21	84.00	23.97	63.84	60.80	20.81	77.33	59.00
DP† (Ratner et al., 2017)	70.96	69.37	82.00	23.78	63.90	64.20	21.12	71.00	58.29
FS† (Fu et al., 2020)	70.36	68.68	76.80	0.00	60.98	31.40	34.3	31.83	46.79
Neural									
WeaSEL (Cachay et al., 2021)	85.16	91.23	96.40	2.94	85.92	64.2	0.00	44.30	58.77
Denoise† (Ren et al., 2020)	76.22	71.56	76.56	91.69	83.45	56.20	22.47	80.83	69.87
KnowMAN (März et al., 2021)	59.00	76.76	94.00	92.80	84.68	65.20	25.48	80.50	72.30
MV+RoBERTa†	85.76	89.91	96.56	94.17	**86.88**	66.28	17.99	84.00	77.69
DP+RoBERTa†	86.26	89.59	95.60	28.25	86.81	72.12	17.62	70.57	68.35
FS+RoBERTa†	**86.95**	**92.08**	93.84	10.72	86.69	30.44	0.0	31.83	54.07
SepLL	83.57	91.32	**97.50**	**95.45**	85.47	**81.25**	**43.2**	**87.33**	**83.14**
Cosine based (Yu et al., 2021)									
MV+Cosine†	**88.22**	94.23	**97.60**	96.67	**88.15**	77.96	40.5	86.20	83.69
DP+Cosine†	87.91	94.09	96.80	31.71	87.53	82.36	28.86	75.17	73.05
SepLL+Cosine	88.00	**95.07**	**97.60**	**97.01**	86.28	**83.40**	**43.37**	**86.83**	**84.70**

Table 2: Results on the Wrench benchmark tasks. Accuracy values are reported for all datasets, except for SMS and Spouse where the binary F_1 is shown due to the large class imbalance. Numbers directly taken from the Wrench paper are marked by †.

4.3 Hyperparameters

Two types of hyperparameters are tuned, namely transformer related hyperparameters and information routing related ones. We perform a grid search on these and take the best model based on the validation set. The first group includes a learning rate in $\{1^{-5}, 2^{-5}, 5^{-5}\}$, a batch size of 16 and warmup steps in $\{0, 100\}$. For information routing related hyperparameters we search for weight decay in $\{0, 0.01, 0.001\}$, L_2-regularization on the LF path in $\{0.1, 0.5, 0., 1.\}$, and for noise injection $\lambda \in \{0., 0.1, 0.2, 0.05\}$. *SepLL* is always trained using AdamW (Loshchilov and Hutter, 2019).

4.4 Implementation and Reproducibility

Our implementation is in JAX (Bradbury et al., 2018), using the Huggingface transformers library (Wolf et al., 2020) as a high level interface.

In order to save resources, we report numbers directly from the Wrench paper, whenever applicable. In other cases, we use the datasets and the evaluation setting of Wrench, and the original implementations and reported hyper-parameters of the respective publications.

We use RoBERTa-base as backbone of most models, so all models use approximately 125 million parameters. Fine-tuning one instance takes between 5 and 60 minutes, depending on the dataset size. The experiments are performed on a Nvidia DGX-1 using a single Tesla V100 graphics card per run.

5 Experiments

This section provides an analysis of the capabilities of the model. After the general performance analysis, an ablation study of the information strategies is performed. Furthermore, we investigate how much information flows through the path associated with the latent class prediction, and how much though the other, LF-specific path.

5.1 General Performance

The results are split into two parts. Firstly, a comparison with the standard baselines is given. Secondly, we analyse the impact of Cosine separately, as we view it as a general framework to self-optimize the prediction performance of any pre-trained classifier. The results are shown in Table 2. *SepLL* outperforms the standard baselines on all tasks except IMDb and AGNews. We think this is due to the fact that both datasets are rather large while having a low number of labeling functions. The same trend is observable when combined with Cosine. On most tasks a new state-of-the-art is reached. A negative exception is given by SemEval, where Cosine is not able to generalize from our base model. Importantly, the average performance over all tasks is improved by a margin of 6% on the standard baselines and 1% in the Cosine section, relative to the respective best-performing

	avg
Full Model	83.14
– Weight decay	82.73
– L_2-reg.	82.59
– Unlabeled data	82.11
– Noise injection	81.77
Basic model	80.85

Table 3: Ablation of information routing strategies. Each strategy is removed individually. The basic model does not use any strategy. The average is taken over all datasets on the test set.

comparison method.

5.2 Ablation of Information Routing Strategies

In order to understand the impact of the routing strategies we perform an ablation study, which is shown in Table 3. Starting from the full model, each routing strategy is removed individually, and in the last row all strategies are removed. The table shows the average performance over all datasets. Given that there are different types of metrics this is just an aggregate view of the performance – a detailed ablation including the performance per task is given in Appendix B.

We observe that each strategy helps the performance of the end model and that the noise injection strategy has the highest positive impact. This reinforces the initial assumption that a sample has a larger learning impact on the task-specific path if multiple labeling functions belonging to one class match simultaneously.

Nevertheless, the basic model (without active routing) performs decent in comparison to the baselines in Table 2. This is surprising because there is no apparent incentive to prevent the model to collapse to the LF path. One possible explanation is that the gradient updates nevertheless flow through both paths and still update the task path in a way that it is consistent with the LF-prediction. Moreover, predicting the LF through the task-path, albeit less accurate, is more parameter efficient than going through the LF-path.

5.3 Labeling Function Memorization

As a by-product of the architecture, it is possible to predict whether a labeling function matches or not. As these matches represent the learning signal, an evaluation of the prediction of labeling function matches also shows how much information is retained. In Figure 2 multiple metrics are computed to analyze the memorization of matches. The accuracy and the F_1-score (because the matrix L is sparse) for LF-predictions, averaged over all test sets, are computed to measure memorization of the labeling functions matches.

In order to transform logits into predictions, we compute the softmax activation on the functions $f_{\tilde{L}}$ and $f_{\hat{L}}$, and define the prediction as

$$L_{ij} = 1 \Leftrightarrow \text{softmax}\left(f_*(z_i)\right)_j > \frac{4}{m}$$

otherwise $L_{ij} = 0$, i.e., if a sample has 4 times the probability of the uniform distribution. To compute the task predictions, we apply the same threshold to the softmax of the mapped task logits, i.e., $\text{softmax}\left(f_{\tilde{Y}}(z_i)T^\top\right)$. We call the outputs of the softmax prediction distribution. The diagram on the right in Figure 2 displays the cross-entropy between the LF distribution P (see section 3) and the prediction distributions, and the uniform distribution $U[0, 1]$.

The results show that the prediction from the final output and from the LF path are nearly identical. Interestingly, the latent LF path achieves a slightly better accuracy but slightly worse F_1-score. In comparison to the full and the latent model the cross-entropy between $\tilde{Y}$ and P is high where the latter approaches the cross-entropy between $\hat{L}$ and the uniform distribution. These results indicate that $\tilde{Y}$ does not substantially influence the prediction of L. Still, the prediction made from the task-related path $f_{\tilde{Y}}$ reaches an average F_1-score of 88.5%, which shows that it still contains much of the information needed to predict LF matches. In appendix C an additional figure shows that the difference between performance on the training set and the test set is low, indicating that there is little to no overfitting towards the training LF distribution.

5.4 Impact of Number of LF Matches

One could expect that it is easier to predict the class label for samples with many labeling functions matches. Figure 3 shows the performance of each task on the test set in relation to the number of labeling function matches (note that we use those LF-matches only for analysis purposes, they are not observed by the model) . The figure does not contain SemEval as it has exactly one match per sample.

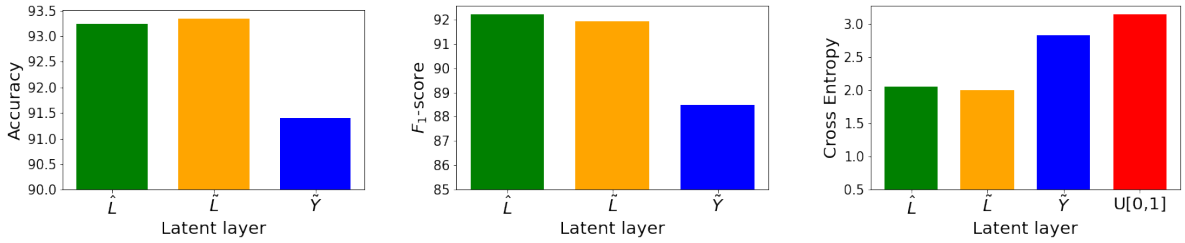

Figure 2: The figure shows the accuracy (left), the F_1-score (center) and the cross-entropy (right) between true labels L and the predictions based on the full model $f_{\hat{L}}$ and the latent representations $f_{\tilde{Y}}, f_{\tilde{L}}$ predictions. On the right an additional comparison to the uniform distribution is presented.

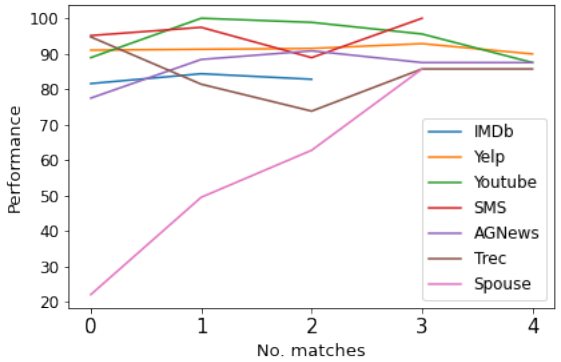

Figure 3: Performance (accuracy or F1 score) on the tasks given the amount of labeling function matches (test set).

Interestingly, the performance on samples with no LF-match is, for most datasets (apart from Spouse), almost on par with samples that include patterns from one or more LFs. It is therefore fair to say that the model generalizes beyond the labeling function information.

An exception is given by the Spouse dataset, which has the lowest coverage in the training set (see Table 1). Appendix D provides additional numbers, including a table which shows the amount of samples per number of LF-matches per dataset.

6 Conclusion

This work tackles weak supervision in a novel way. Instead of denoising the weak labels or iteratively updating them, the weak labels are used as the training target as-is. We introduce the model *SepLL*, which separates information relevant for the target task through the usage of two latent spaces. One latent space is used to perform downstream task prediction, the other one aims to keep the labeling function specific information. Thus, the prediction is made from the latent space without any direct supervision. Experiments show that the model is able to achieve state-of-the-art performance. An additional investigation shows that the model is able to memorize the labeling function information. Hence, the model cannot only be used for downstream tasks but also to predict labeling function matches.

7 Limitations

As in the experiments in the Wrench benchmark, we use validation data for early stopping. If a setting was purely weakly supervised, with no annotated data, this would not be possible. To ensure comparability, we stick to the Wrench setup, which assumes that the cost of annotating a small sample size for validation is acceptable in most scenarios.

Weak supervision performance highly depends on the quality and other properties of the labeling functions. In contrast to more standardized scenarios of machine learning, e.g. in supervised learning, where it can be assumed that training and test data samples are drawn i.i.d. from the same distribution, such arguments cannot be made about the labeling functions and their relationship to the correct annotations.

Formal analysis, as it has been attempted sometimes for weak supervision, relies on heavy assumptions, which are typically unrealistic and likely to be wrong. Such assumptions could be: the weak labelers produce noise following a defined noise distribution; the weak annotations are independent given the class label; each weak labeler exceeds a threshold accuracy; etc. In this paper, we do not attempt a formal analysis based on such assumptions. However, we compare to other methods which are based on such assumptions, and we outperform them in our experiments. With other data sets, potentially showing other characteristics (other tasks,

other languages, other labeling functions), the performance could be different. In particular, since the weakly supervised method performs almost as well as the supervised model, there is a need for tasks and datasets that are more challenging.

Another limitation is that currently our model is only implemented for classification, not for sequence tagging. This adaptation would be possible, but is not trivial since sequential dependencies, unlabeled tokens, search, etc. would need to be carefully handled. We leave these extensions for future work.

References

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.

Tiago A Almeida, J M G Hidalgo, and A Yamakami. 2011. Contributions to the study of SMS spam filtering: new collection and results. In *DocEng '11*.

Abhijeet Awasthi, Sabyasachi Ghosh, Rasna Goyal, and Sunita Sarawagi. 2020. Learning from Rules Generalizing Labeled Exemplars.

Y Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation Learning: A Review and New Perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35:1798–1828.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. 2018. JAX: composable transformations of Python+NumPy programs.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners.

Salva Rühling Cachay, Benedikt Boecking, and Artur Dubrawski. 2021. End-to-End Weak Supervision. In *Advances in Neural Information Processing Systems*.

Pierre Comon. 1994. Independent component analysis, A new concept? *Signal Processing*, 36(3):287–314.

D Corney, M-Dyaa Albakour, Miguel Martinez-Alvarez, and Samir Moussa. 2016. What do a Million News Articles Look like? In *NewsIR@ECIR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Daniel Y. Fu, Mayee F. Chen, Frederic Sala, Sarah M. Hooper, Kayvon Fatahalian, and Christopher Ré. 2020. Fast and three-rious: Speeding up weak supervision with triplet methods. In *ICML*, pages 3280–3291.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. SemEval-2010 Task 8: Multi-Way Classification of Semantic Relations between Pairs of Nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38, Uppsala, Sweden. Association for Computational Linguistics.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P Burgess, Xavier Glorot, Matthew M Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. In *ICLR*.

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.

Giannis Karamanolakis, Subhabrata Mukherjee, Guoqing Zheng, and Ahmed Hassan Awadallah. 2021. Self-Training with Weak Supervision.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Diederik P Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Ro{BERT}a: A Robustly Optimized {BERT} Pretraining Approach.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

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.

Luisa März, Ehsaneddin Asgari, Fabienne Braune, Franziska Zimmermann, and Benjamin Roth. 2021. KnowMAN: Weakly supervised multinomial adversarial networks. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9549–9557, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

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.

Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. *Advances in neural information processing systems*, 29.

Wendi Ren, Yinghao Li, Hanting Su, David Kartchner, Cassie Mitchell, and Chao Zhang. 2020. Denoising Multi-Source Weak Supervision for Neural Text Classification. *Findings of the Association for Computational Linguistics: EMNLP 2020*.

Sebastian Ruder, Matthew E Peters, Swabha Swayamdipta, and Thomas Wolf. 2019. Transfer Learning in Natural Language Processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18, Minneapolis, Minnesota. Association for Computational Linguistics.

Anastasiia Sedova, Andreas Stephan, Marina Speranskaya, and Benjamin Roth. 2021. Knodle: Modular Weakly Supervised Learning with PyTorch. In *Proceedings of the 6th Workshop on Representation Learning for NLP (RepL4NLP-2021)*, pages 100–111, Online. Association for Computational Linguistics.

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.

Julius von Kügelgen, Yash Sharma, Luigi Gresele, Wieland Brendel, Bernhard Schölkopf, Michel Besserve, and Francesco Locatello. 2021. Self-Supervised Learning with Data Augmentations Provably Isolates Content from Style.

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.

Yue Yu, Simiao Zuo, Haoming Jiang, Wendi Ren, Tuo Zhao, and Chao Zhang. 2021. Fine-Tuning Pre-trained Language Model with Weak Supervision: A Contrastive-Regularized Self-Training Approach.

Jieyu Zhang, Yue Yu, Yinghao Li, Yujing Wang, Yaming Yang, Mao Yang, and Alexander Ratner. 2021. WRENCH: A Comprehensive Benchmark for Weak Supervision.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

Dawei Zhu, Xiaoyu Shen, Michael A Hedderich, and Dietrich Klakow. 2022. Meta Self-Refinement for Robust Learning with Weak Supervision. *arXiv preprint arXiv:2205.07290*.

Roland S Zimmermann, Yash Sharma, Steffen Schneider, Matthias Bethge, and Wieland Brendel. 2021. Contrastive Learning Inverts the Data Generating Process. In *ICML*.

A Experimental Description

As mentioned in the main paper, the code, setup and hyperparameters for the baselines are taken from the Wrench benchmark and are described in the corresponding paper (Zhang et al., 2021). For the Gold+RoBERTa model there was no result available, as there are no ground truth labels for train set. Thus we assume an F_1-score of 45 for the computation of the average as it is better than the best performing model.

In the following paragraphs, the hyperparameters for the baselines we trained ourselves are briefly presented.

Weasel (Cachay et al., 2021). We experiment with hyperparameters similar to the original paper because there no transformer encoder "roberta-base" is used. We use AdamW (Loshchilov and Hutter, 2019) and optimize hyperparameters for temperature in $\{0.33, 1.0\}$, dropout in $\{0.1, 0.2\}$,

weight decay in $\{1^{-4}, 7^{-7}\}$ and learning rate in $\{1^{-4}, 1^{-5}, 2^{-5}\}$. Unfortunately we were unable to obtain reasonable performance on the skewed datasets SMS and Spouse. The original paper does not use pre-trained language models, thus no direct comparison is possible. We decided to take the same encoder for all models.

KnowMAN (März et al., 2021). The values are set similar to the original paper. Batch size is 16, hidden size 700, dropout is 0.4 and trained is up to 5 epochs. The classifier(C) and the discriminator(D) use Adam (Kingma and Ba, 2014) with a learning rate of $1^{-}4$ and the feature extractor(F) uses AdamW (Loshchilov and Hutter, 2019) with the same learning rate. For all three, *num_layer* is set to 1.

B Ablations

In addition to the summarized ablation in section 5.2, Table 4 shows the impact of the ablations on task level. We observe that in general, the results for each task agree with the average.

C Labeling Function Memorization

The detailed numbers corresponding to the evaluation of section 5.3 are given in Table 5. The absolute difference between train and test split is shown in Figure 4. The differences are low for accuracy, F_1-score and cross-entropy, thus we conclude that there is no or rather little overfitting towards the train set.

D Impact of number of matches

Table 5 provides detailed numbers for Figure 3. Since Figure 3 does not reflect how many samples correspond to a certain number of LF matches, detailed counts are added in Table 7. This is an important addition as a performance measurement on a small number of samples is not indicative of the true performance. Luckiliy, usually the number of matching samples per number of matching labeling functions is distributed rather well.

	IMDb	Yelp	Youtube	SMS	AGNews	Trec	Spouse	SemEval	Avg.
Full model	83.57	91.32	97.50	95.45	85.47	81.25	43.20	87.33	83.14
– Weight decay	83.57	91.32	97.50	95.45	85.47	79.03	42.55	86.99	82.73
– L_2 reg.	83.25	91.14	97.08	94.57	85.47	79.03	43.20	86.99	82.59
– Unlabeled data	82.13	88.34	97.50	95.45	85.32	81.25	39.89	86.99	82.11
– Noise injection	82.13	88.34	97.50	95.45	85.46	78.43	39.89	86.99	81.77
Basic model	82.13	86.89	97.08	93.85	85.14	74.80	39.89	86.99	80.85

Table 4: Extension of the ablation in Table 3, showing all datasets.

Dataset	$\tilde{L}$ Acc.	$\hat{L}$ Acc.	$\tilde{Y}$ Acc.	$\tilde{L}$ F_1	$\hat{L}$ F_1	$\tilde{Y}$ F_1	$\tilde{L}$ CE	$\hat{L}$ CE	$\tilde{Y}$ CE	$U[0,1]$ CE
IMDb	90.21	90.24	86.88	89.41	89.58	80.77	1.42	1.41	2.11	2.20
Yelp	91.44	91.46	89.94	89.46	89.51	85.18	2.19	2.13	2.58	2.71
Youtube	82.80	83.00	81.60	76.32	77.03	73.33	1.72	1.70	2.16	2.30
SMS	96.41	95.65	99.28	97.59	97.19	98.92	4.26	5.31	4.76	4.29
AGNews	91.05	91.54	89.48	89.06	90.45	84.50	1.72	1.66	1.92	2.20
Trec	99.42	99.23	95.49	99.42	99.24	95.90	1.11	1.05	3.59	4.22
Spouse	95.98	96.32	95.81	94.64	95.89	93.76	2.03	2.01	2.14	2.20
SemEval	99.47	98.45	92.70	99.52	98.81	95.55	1.55	1.21	3.36	5.10
Avg.	93.35	93.24	91.40	91.93	92.21	88.49	2.00	2.06	2.83	3.15

Table 5: Extension of Figure 2, showing all numbers explicitly. The presented metrics are accuracy, macro F_1-score and cross entropy (CE) between true LF matrix L (see section 3) and the rows which are the full model $f_{\hat{L}}$, the latent layers $f_{\tilde{L}}, f_{\tilde{Y}}$ and the uniform distribution $\mathcal{U}[0,1]$.

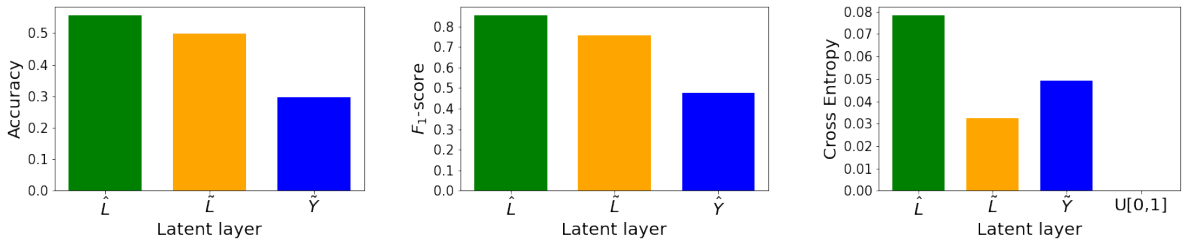

Figure 4: This is extension to Figure 2, describing the same metrics. But here we compute the the absolute difference between the metrics computed on the train set and the test set.

	0	1	2	3	4	5
IMDb	81.58	84.35	82.80	0.00	0.00	0.0
Yelp	91.00	91.25	91.48	92.84	89.93	62.5
Youtube	88.89	100.00	98.84	95.56	87.50	0.0
SMS	95.12	97.44	88.89	100.00	0.00	0.0
AGNews	77.48	88.38	90.81	87.50	87.50	0.0
Trec	94.74	81.40	73.81	85.71	85.71	0.0
Spouse	22.06	49.51	62.77	85.71	0.00	0.0
SemEval	0.00	85.09	0.00	0.00	0.00	0.0

Table 6: Detailed numbers corresponding to Figure 3. The columns describe the number of matches a sample has, the cells the performance, i.e. accuracy or F_1-score.

#matches	total	0	1	2	3	4	5
IMDb	2953	304	1521	593	0	0	0
Yelp	5732	611	1463	1092	475	139	16
Youtube	460	18	86	86	45	8	0
SMS	263	302	148	37	12	0	0
AGNews	11367	3699	5622	2318	336	24	0
Trec	788	19	301	84	70	21	0
Spouse	1019	1951	519	196	33	1	0
SemEval	764.0	0	436	0.0	0.0	0.0	0.0

Table 7: It is shown how many samples there are in total and split up in number of matches per sample. Numbers are computed on the test set.