Automatic Rule Induction for Efficient Semi-Supervised Learning

Reid Pryzant, Ziyi Yang, Yichong Xu, Chenguang Zhu, Michael Zeng
Microsoft Cognitive Services Research Group
{reidpryzant,ziyiyang,yicxu,chezhu,nzeng}@microsoft.com

Abstract

Semi-supervised learning has shown promise in allowing NLP models to generalize from small amounts of labeled data. Meanwhile, pretrained transformer models act as black-box correlation engines that are difficult to explain and sometimes behave unreliably. In this paper, we propose tackling both of these challenges via Automatic Rule Induction (ARI), a simple and general-purpose framework for the automatic discovery and integration of symbolic rules into pretrained transformer models. First, we extract weak symbolic rules from low-capacity machine learning models trained on small amounts of labeled data. Next, we use an attention mechanism to integrate these rules into high-capacity pretrained transformer models. Last, the rule-augmented system becomes part of a self-training framework to boost supervision signal on unlabeled data. These steps can be layered beneath a variety of existing weak supervision and semi-supervised NLP algorithms in order to improve performance and interpretability. Experiments across nine sequence classification and relation extraction tasks suggest that ARI can improve state-of-the-art methods with no manual effort and minimal computational overhead.

1 Introduction

Large-scale pretrained neural networks can struggle to generalize from small amounts of labeled data (Devlin et al., 2019), motivating approaches that leverage both labeled and unlabeled data. This is partially due to the black-box and correlational nature of neural networks, which confers the additional difficulties of uninterpretability (Bolukbasi et al., 2021) and unreliability (Sagawa et al., 2020).

A growing body of research seeks to ameliorate these issues by augmenting neural networks with symbolic components: heuristics, logical formulas, program traces, network templating, blacklists, etc (Arabshahi et al., 2018; Galassi et al., 2020; Wang et al., 2021). In this paper, we refer to these components as *rules*. Symbolic reasoning has attractive properties. Rules need little or no data to systematically generalize, and rules are inherently interpretable with respect to their constituent operations.

In this paper we propose a general-purpose framework for the automatic discovery and integration of symbolic rules into pretrained models. The framework contrasts with prior neuro-symbolic NLP research in two ways. First, we present a fully automatic rule generation procedure, whereas prior work has largely focused on manually crafted rules (Mekala and Shang, 2020; Awasthi et al., 2020; Li et al., 2021) or semi-manual rule generation procedures (Boecking et al., 2020; Galhotra et al., 2021; Zhang et al., 2022). With these existing techniques, practioners must formulate and implement their rules by hand, creating a second-order "rule annotation" burden on top of the data labeling process.

Second, the proposed framework is general purpose and can be applied to any classification dataset. This contrasts with prior research that proposes task- and domain-specific symbolic logic, through weak supervision signals (Ratner et al., 2017; Awasthi et al., 2020; Safranchik et al., 2020), special loss functions (Xu et al., 2018), model architectures (Seo et al., 2021), and prompt templates (Schick and Schütze, 2020a).

Our framework consists of two steps. First, we generate symbolic rules from data. This involves training low-capacity machine learning models on a reduced feature space, extracting artifacts from these models which are predictive of the class labels, then converting these artifacts into rules. After the rule induction step, we use the induced rules to amplify training signal in the unlabeled data. In particular, we adopt a rule-augmented self-training procedure, using an attention mechanism to aggregate the predictions of a backbone classifier (e.g. BERT) and the rules.

We evaluate the ARI framework across nine text

classification and relation extraction tasks. The results suggest that the proposed algorithm can exceed state-of-the-art semi-supervised baselines, and that these gains may be because the model learns to rely more heavily on rules for difficult-to-predict examples. We also show that the proposed rule induction strategy can rival human crafted rules in terms of their quality. Last, we demonstrate the interpretabiltiy of the overall system. In summary, the contributions of this paper are:[1]

- Methods for automatically inducing and filtering symbolic rules from data.
- A self-training algorithm and attention mechanism for incorporating these rules into pre-trained NLP models.
- Evidence suggesting the proposed framework can be layered beneath a number existing algorithms to boost performance and interpretability.

2 The ARI Framework

The proposed rule induction framework seeks to automatically induce symbolic rules from labeled data. Next, the rules can be used to amplify training signal on the unlabeled data. These steps are depicted in Fig. 1.

More formally, assume we are given a target classification task consisting of labeled classification data $\mathcal{L} = \{(x_i, y_i)\}_{i=1}^{M}$ and unlabeled data $\mathcal{U} = \{(x_{i+M})\}_{i=1}^{N}$, where each x_i is a text string and $y_i \in \{1, ..., K\}$. Our proposed method uses the labeled data $\mathcal{L}$ to generate a set of symbolic prediction functions ("rules") $\mathcal{R} = \{r_j\}_{j=1}^{R}$ that take the text and output a label or abstain: $r_j(x) \in \{-1\} \cup \{1, ..., K\}$. We then train a joint system which models $P(y|x;\ \mathcal{L}, \mathcal{U}, \mathcal{R})$, i.e., an estimator which utilizes the labeled data, unlabeled data, and rules to make reliable and interpretable predictions.

2.1 Rule Induction

We begin by explaining our rule induction technique. Concretely, the goal is to generate a set of prediction functions which use the text to output a label or abstain. We operationalize this as a three-stage pipeline. First, we featurize the text. Second, we use these features to construct rule-based predictor functions. Last, we filter the rules in order

[1]An open-source implementation of the framework is available at: https://github.com/microsoft/automatic-rule-induction.

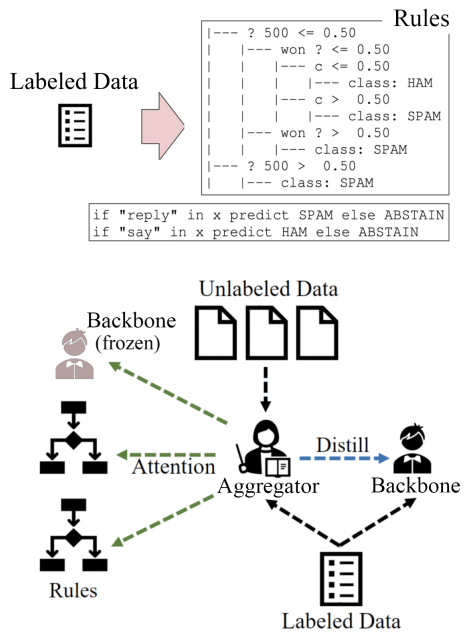


Figure 1: Overview of the proposed Automatic Rule Induction (ARI) framework. First, rules are induced from labeled data (top, shown with real example rules). Second, the rules are integrated into pre-trained NLP models via an attention mechanism and a self-training procedure (bottom).

to block them from firing on risky examples (to maximise precision).

Text Featurization. In the first step, the input text x_j is converted into a binary or continuous feature space $\phi(x_j) \in \mathbb{R}^d$ that is more amenable to symbolic reasoning than the raw text.

1. *Ngram* (ϕ^N). We adopt a bag-of-words model of the text, converting each string into a binary vector reflecting the presence or absence of words in a vocabulary of size V.

2. *PCA* (ϕ^P). Intuitively, if we only have a small amount of labeled data, then common ngrams may be spuriously correlated with the labels. To tackle this issue, we follow Arora et al. (2017); Yang et al. (2021) by subtracting off a vector of shared information from each feature matrix. Specifically, we compute the first principal component v of an ngram feature matrix $P \in \mathbb{R}^{(M+N)\times d}$ constructed from both labeled and unlabeled texts in a dataset, i.e., the jth row $P_{j,:} = \phi^N(x_j) : j \in [1, M+N]$. Then it follows that singular value decomposition (SVD) of the ngram feature matrix is $P = U\Sigma V^T$. The first principal component v is the most "common" part of all samples

(e.g., common words), and is defined as the first column of $V \in \mathbb{R}^{d \times d}$. We then remove the projection of all features vectors $\{\phi^N(x)\}$ onto v:

$$\phi^P(x) := \phi^N(x) - v \frac{v^T \phi^N(x)}{\|\phi^N(x)\|_2}$$

We hypothesize that this can help remove common information that is shared across many texts, in order to isolate the most unique and salient lexical phenomena.

Rule Discovery. Armed with a featurization of the texts in $\mathcal{L}$, we proceed by generating symbolic rules from the features which are capable of predicting the labels with high precision. In practice, these rules are artifacts of low-capacity machine learning models. We experiment with two rule generation algorithms.

The first rule generation algorithm uses a *linear* model and can be applied to ngram-based (binary) feature spaces. It involves training a simple linear model $m(x_j) = \sigma(\mathbf{W}\phi(x_j))$ containing one matrix of parameters $\mathbf{W} \in \mathcal{R}^{K \times V}$ that predicts class labels from the input features. It is trained by using a cross-entropy loss function and l_2 regularization term (Tibshirani, 1996). Note that in this case σ represents an element-wise sigmoid function (Mao et al., 2014). Next, we select the R largest weights in $\mathbf{W}$ and create one rule from each weight. If a selected weight $w_{i,k}$ corresponds to feature f_i and label k, then we create a rule r that predicts label k if the ith dimension of $\phi(x_j)$ is 1, otherwise abstaining:

$$r(x_j) = \begin{cases} k & \text{if } \phi_i(x_j) = 1 \\ -1 & \text{otherwise} \end{cases}$$

The second rule generation algorithm uses decision trees and can be applied to ngram- or PCA-based (binary or continuous) feature spaces. Intuitively, we want to find regions inside the range of each feature (or combination of features) that are predictive of the labels. We accomplish this by training a random forest classifier containing R decision trees at a depth of D (we use $D = 3$ in the experiments). To make a rule from each decision tree, we apply a confidence threshold τ to the predicted label distribution in order to control the boundary between prediction and abstainment. In other words, if a decision tree t_i outputs a probability distribution $\hat{p}$ over the labels, i.e. $t_i(\phi(x_j)) = \hat{p}_{i,j}$ then we construct a rule r_i such that:

$$r_i(x_j) = \begin{cases} \arg\max(\hat{p}_{i,j}) & \text{if } \max(\hat{p}_{i,j}) > \tau \\ -1 & \text{otherwise} \end{cases}$$

Note that due to the bagged construction of the random forest, we hypothesize that these decision trees will yield rules which can be aggregated for robust supervision signal.

Rule Filtering. Since rules are allowed to abstain from making predictions, we can introduce dynamic filtering mechanisms that block rules from firing on examples where the rule is likely to make errors. This helps increase the precision of our rules and increase the fidelity of our downstream rule integration activities.

- *Training accuracy.* The rules are not perfect predictors and can make errors on the training set. We randomly sample a proportion of these errors (50% in the experiments) and replace the incorrectly predicted value with abstainment (-1).

- *Semantic coverage.* We design a filter to ensure that the "covered" subset of examples (examples where at least one rule fires) resembles the training set. In detail, after a rule r_i fires on input text x_j, predicting label $r_i(x_j) = l$, we use the Sentence BERT framework (Reimers and Gurevych, 2019) and a pre-trained mpnet model (Song et al., 2020) to obtain embeddings for the input sentence x_j and all training samples that have the same label as the rule's prediction: $\{x_i \in \mathcal{L} \ : \ y_i = l\}$. We then compute the cosine similarity between the input's embedding and the training set embeddings. If the maximum of these similarities is below some threshold (0.8 in the experiments) then we block the rule r_i from firing and replace its prediction l with abstainment (-1).[2]

2.2 Rule Integration

After we have induced weak symbolic rules $\{r_i\}_{i=1}^{R}$ from the labeled data $\mathcal{L}$, we can leverage the rules and unlabeled data $\mathcal{U}$ for extra training signal.

[2]Note that in applied settings, this may be computed on the fly with fast similarity search packages, e.g. Johnson et al. (2019). For this initial work, we pre-computed all filters prior to model training.

Our method is inspired by recent work in weak supervision and semi-supervised learning (Karamanolakis et al., 2021; Du et al., 2020). It consists of a backbone classification model (e.g. BERT) and a proposed rule aggregation layer. The aggregation layer uses an attention mechanism to combine the outputs of the backbone model and rules. The parameters of the backbone and aggregator are jointly trained via a self-training procedure over the labeled and unlabeled data.

In more detail, the **backbone** model $b(\cdot)$ is a standard BERT-based classifier with a prediction head attached to the `[CLS]` embedding. This classifier outputs a probability distribution over the possible labels.

The **aggregation** layer $a(\cdot)$ is trained to optimally combine the predictions of the backbone model and rules. It does so via the following attention mechanism. The layer first initializes trainable embeddings e_j for each rule r_j, and embedding e_s for the backbone. Next, it computes dot-product attention scores between these embeddings and an embedded version of the input text (h_i). The final model prediction is a weighted sum of the backbone and rule predictions, where the weights are determined by the attention scores.

Specifically, if the set of rules activated on input x_i is $R_i = \{r_j \in \mathcal{R} : r_j(x_i) \neq -1\}$, and the function $g(\cdot) \in \mathcal{R}^K$ returns a one-hot encoding of its input, then the rule aggregation layer computes a probability distribution over the labels:

$$a(x_i) = \frac{1}{Q}\left(\sum_{j:r_j \in R_i} s_i^j \, g(r_j(x_i)) + s_i^s \, b(x_i) + u\right) \tag{1}$$

where the attention scores are calculated as,

$$s_i^j = \sigma(p(h_i) \cdot e_j)$$

Note that p is a multi-layer perceptron that projects the input representation h_i into a shared embedding space, Q is a normalizing factor to ensure $a(x_i)$ is a probability distribution, $\sigma(\cdot)$ is the sigmoid function. Following Karamanolakis et al. (2021), the quantity u is a uniform smoothing term.

In order to train the overall system, we first pretrain the backbone on the labeled data $\mathcal{L}$. Next we iteratively co-train the backbone and aggregation layer. We train the aggregator (freezing the parameters of the backbone), then train the backbone (freezing the aggregator). The process is as follows:

1. Train the backbone s using labeled data $\mathcal{L}$ and a cross-entropy loss function, where $b(x_i)_{y_i}$ denotes the logit for the groundtruth class y_i:

$$\ell_{stu}^{sup} = -\sum_{(x_i,y_i)\in\mathcal{L}} \log b(x_i)_{y_i}$$

2. Repeat until convergence:

   (a) Train the aggregator t on labeled data using a cross-entropy loss function :

$$\ell_{tea}^{sup} = -\sum_{(x_i,y_i)\in\mathcal{L}} \log a(x_i)_{y_i}$$

   (b) Train the aggregator on unlabeled data $\mathcal{U}$ with a minimum entropy objective (Grandvalet and Bengio, 2004). This encourages the aggregator to learn attention scores that favor rule agreement, because the aggregator will be encouraged to output more focused probability distributions, thereby placing less importance on spurious rules that disagree:

$$\ell_{tea}^{unsup} = -\sum_{x_i\in\mathcal{U}} a(x_i)^T \log a(x_i)$$

   where $\log a(x_i) \in \mathbb{R}^K$ denotes the element-wise logarithm of the probability distribution $a(x_i)$.

   (c) Train the backbone on labeled data using ℓ_{stu}^{sup}:

$$\ell_{stu}^{sup} = -\sum_{(x_i,y_i)\in\mathcal{L}} \log b(x_i)_{y_i}$$

   (d) Train the backbone on unlabeled data by distilling from the aggregator, i.e. train the backbone to mimic the aggregator's output:

$$\ell_{stu}^{unsup} = -\sum_{x_i\in\mathcal{U}} a(x_i)^T \log b(x_i)$$

Once trained, one can use the outputs of either the backbone or aggregator for inference. If one uses the aggregator, they receive the benefit of improved interpretability: one could inspect the attention scores s_i^j to understand what proportion of the system's decision was due to each rule.[3]

[3] Recent research shows that attention distributions in hidden layers are not valid explanations (Wiegreffe and Pinter, 2019), however in our case the attention scores are part of the model's output layer, i.e. used in a linear combination to calculate output probabilities directly. See Section 4 for details.

	AGNews	CDR	ChemProt	IMDB	SciCite	SemEval	SMS	TREC	Youtube
Domain	News	Bio	Bio	Review	CS	Web	Sms	Speech	Web
# Labeled	4800	421	643	1000	412	87	228	248	79
# Unlabeled	91200	8009	12218	19000	7831	1662	4343	4717	1507
# Valid	1500	920	1500	1500	916	178	500	500	120
# Test	12000	4673	1607	2500	1861	600	500	500	250
# Classes	4	2	10	2	3	9	2	6	2

Table 1: Datasets used in our experiments.

3 Experiments

We perform experiments across 9 datasets and tasks, finding that the ARI rule induction framework can improve the performance of state-of-the-art semi-supervised text classification algorithms. Note that concrete examples of the human-readable rules succeeding (and failing) are given in the Appendix.

3.1 Experimental Setup

We evaluate our framework on nine benchmark NLP classification datasets that are popular in the few-shot learning and weak supervision literature (Ratner et al., 2017; Awasthi et al., 2020; Zhang et al., 2021a; Cohan et al., 2019). These tasks are as follows: **AGNews**: using news headlines to predict article topic, **CDR**: using scientific paper excerpts to predict whether drugs induce diseases, **ChemProt**: using paper experts to predict the functional relationship between chemicals and proteins, **IMDB**: movie review sentiment, **SciCite**: classifying citation intent in Computer Science papers, **SemEval**: relation classification from web text, **SMS**: text message spam detection, **TREC**: conversational question intent classification, **Youtube**: internet comment spam detection.

Table 1 shows dataset statistics. Our benchmarks cover a range of discourse domains and classification types. Unless otherwise stated we consider a 5% / 95% split between labeled data and unlabeled data. We construct this split by randomly partitioning the total training data and removing labels from the 95% split. Following Gao et al. (2020); Zhang et al. (2022) we subsample each validation set so that it roughly matches the size of the training set in order to better simulate label scarcity.

All reported results are the average of ten experimental trials, each with different random splits, seeds, and initializations. For each trial, we continuously train our models for 12,500 steps using a batch size of 32, and we stop the training process early based on validation set performance. For each method (baseline and proposed), we conducted a minimal hyperparameter search (details in the Appendix) to establish the best validation performance before running inference over the test set. We ran all experiments on Microsoft Azure cloud compute using NVIDIA V100 GPUs (32G VRAM). All algorithms were implemented using the Pytorch and Wrench frameworks (Paszke et al., 2017; Zhang et al., 2021a). We report binary F1 score for binary classification tasks and macro-weighted F1 for multiclass classification tasks.

3.2 Baselines

We experiment with our **ngram** and **pca**-style featurization schemes, as well as our linear model (**linear**) and decision tree (**tree**)-based rule generation methods. We compare against the following baselines:

BERT: directly fine-tuning a BERT model on the available supervised data (Devlin et al., 2019). **Weak Ensemble**: It is possible that traditional ML models like regressions and decision trees achieve good performance in these low-resource settings, and the proposed ARI framework just takes advantage of these models. We accordingly train several weak models (BERT, regression, and random forest using the same hyperparameters as was used to obtain rules) and ensemble their predictions for comparison. **LMFT**: training a BERT model on the unlabeled data with its original language modeling objective before fine-tuning on the supervised data (Howard and Ruder, 2018; Gururangan et al., 2020). **Self-Train**: iteratively self-training towards the predictions of a frozen model on the unlabeled data (Nigam and Ghani, 2000; Lee et al., 2013). **Snuba**: We use the Snuba algorithm (Varma and Ré, 2018) to automatically generate weak labels over the unlabeled data, then a generative label

Methods	AGNews	CDR	ChemProt	IMDB	SciCite	SemEval	SMS	TREC	Youtube	Avg.
Baselines										
BERT	90.61	54.92	58.46	87.46	81.88	59.76	95.13	83.38	93.28	76.72
Weak Ensemble	83.76	45.12	41.98	83.46	63.97	48.23	79.33	61.68	88.65	66.24
LMFT	90.59	54.19	58.38	87.48	82.21	60.68	95.59	86.52	93.45	77.13
Self-Train	91.30	55.58	54.90	88.76	81.15	**68.1**	94.99	87.07	93.41	77.89
Snuba	90.46	53.99	58.27	87.29	82.03	60.57	95.43	86.35	93.30	76.98
Min Entropy	90.97	55.14	56.00	89.05	82.01	63.70	95.03	84.92	93.14	77.27
MoE	89.94	54.44	57.27	87.25	81.83	60.93	95.03	85.24	92.53	76.65
VAT	91.31	56.01	55.70	88.49	81.12	*67.99*	95.56	**88.27**	93.19	78.19
PET	**91.46**	51.16	53.90	88.49	75.13	64.45	*95.68*	83.42	**95.36**	75.76
ARI (proposed)										
Ngram + linear	*91.37*	56.11	*60.08*	*89.10*	81.51	65.56	95.6	85.43	*95.24*	78.56
Ngram + tree	91.11	*57.77*	**60.95**	**89.41**	*82.76*	64.01	92.87	86.99	93.55	*78.62*
PCA + tree	90.87	**57.92**	60.01	88.33	**83.76**	65.10	**95.74**	*87.19*	94.95	**79.05**
Oracles										
ASTRA	91.71	61.63	59.58	88.98	82.29	75.18	93.15	87.23	96.42	80.70
T-few	95.12	57.23	55.84	94.48	84.21	64.9	96.73	89.13	96.84	79.79

Table 2: Semi-supervised learning performance on nine classification datasets. Following Tatiana and Valentin (2021), we report the geometric mean in the "Avg." column. We denote the highest and second-highest performance (excluding the expert rules model) in **bold** and *italic* respectively. Note that T-Few was pretrained on AGNews, TREC, and IMDB. See Table 11 for accuracies and comparison against the PR-BOOST algorithm (Zhang et al., 2022).

model from Snorkel (Ratner et al., 2017) to expand the available training data prior to BERT fine-tuning. This baseline offers a direct comparison against a popular weak supervision procedure. **Min Entropy**: Multitask self-training with a minimum entropy objective on the unlabeled data (Grandvalet and Bengio, 2004). **MoE**: This is the same as the model proposed in Section 2.2 except the rules are replaced with two-layer neural network classifiers that are trained end-to-end with the rest of the system. This baseline tests whether the proposed training procedure has the potential of achieving higher accuracy without the rule induction step. This baseline is similar to having a Mixture of Experts layer at the output (Jacobs et al., 1991; Shazeer et al., 2017) without input routing, expert gating, or load balancing. **VAT**: Multitask self-training with a virtual adversarial regularization penalty on the unlabeled data (Miyato et al., 2018). **PET**: a state-of-the-art method for semi-supervised learning that leverages prompting and model ensembling (Schick and Schütze, 2020a). Note that PET is not a fully automatic procedure as it requires prompt templates and class verbilizations for each dataset. We used domain intuition to verbalize each class label, and constructed two prompt templates for each task: "`[MASK] : [example]`" and "`[MASK] : [domain word]: [example]`" where `[domain word]` is a word that signifies the nature of the ensuing text (e.g. "`Review`" for the IMDB dataset). See Appendix for details.

We also compare against two oracles. The first called **ASTRA** and is a state-of-the-art weak supervision algorithm that uses manually designed rules and an iterative self-training procedure (Karamanolakis et al., 2021). For this oracle we use previously published heuristic labeling functions from the weak supervision literature (Zhang et al., 2021a). The rules were manually constructed using domain expertise and, being expertly crafted, suggest an upper bound on performance. The second oracle is called **T-Few** (Liu et al., 2022) and represents a state-of-the-art prompting approach using a large 3 billion parameter model (30 times larger than the rest of the models considered in this paper).

3.3 Experiment Results

Overall results. Table 2 presents our main results. The proposed ARI framework achieves the best performance on 5 out of 9 datasets, and the ARI variations beat the baselines in terms of average performance. Our results suggest that LMFT does not always improve the performance over standard BERT finetuning, and can hurt the performance sometimes (CDR). This is in line with previous research findings (Vu et al., 2021; Du et al., 2020). Self-Train achieves an overall better performance than BERT, but underperformed on ChemProt and overperformed on SemEval. PET

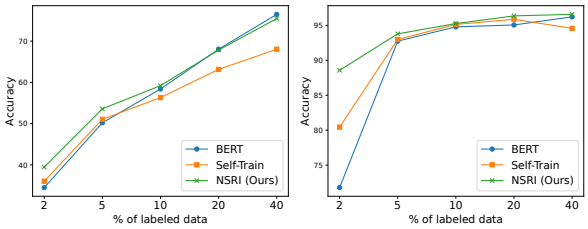
achieves strong results on AGNews and Youtube, but fails on many other datasets. This might be due to its sensitivity to prompts and label words for the scientific domains, which is typical for prompt-based models (Gao et al., 2020). Additionally, due to implementation differences in this prior work, we tested PET after a fixed number of training steps instead of the early-stopping validation technique employed by the other algorithms (Section 3.1).

For ARI, decision-tree based methods give the best results overall, while there is no clear winner between PCA and Ngram-based models. Considering that we also removed stop words in the Ngram features, using PCA to remove common components might not make a big difference to the rules. The performance of ARI is close to ASTRA which uses manually crafted expert rules, showing the potential of automatic rules. Surprisingly, ARI is better than ASTRA on SciCite and SMS by a non-trivial margin. This suggests that automatic rules have the potential to rival human-generated rules. See the Appendix for further results and analysis.

Our results suggest that for prompting methods like PET and T-Few to outperform ARI, one needs bigger models with more language capacity like the 3B parameter Tfew (30x larger than e.g. BERT). ARI and PET leverage smaller models which are faster with reduced memory but also reduced capacity and therefore less effective prompts as prior research as noted (Liu et al., 2021). In the Appendix, we observe that ARI continues to outperform PET when more powerful backbone encoders like DeBERTaV3 (He et al., 2021) are used (Table 10).

Robustness We further test our method's robustness to the number of labeled examples in Fig. 2. We vary the fraction of labeled data between 2% to 40% on the ChemProt and Youtube datasets. The results suggest that ARI can reliably outperform the baselines across this range, especially when labeled data is scarce. Standard supervised BERT fine-tuning become increasingly competitive as the fraction of labeled data exceeds 40%.

Filter Ablations. We provide ablation results on rule filtering methods in Table 3. We pick the best performers between the three rule-generation methods in Table 2 and then vary the filters. All the three methods show performance gains when applied individually, and combining the filters appears to further improve performance in some cases.



(a) ChemProt (b) Youtube

Figure 2: Robustness to training data size.

Filters	CDR	SciCite	SemEval	SMS
None	54.86	81.14	63.67	94.72
Semantic	**57.20**	82.00	65.43	94.54
Train	56.84	81.05	64.36	95.21
Sem. + Train	56.46	**82.41**	**65.56**	**95.74**

Table 3: Ablation study on rule filtering methods. We show the best result between Ngram+linear, Ngram+tree and PCA+tree.

Hard or Soft Labels? There are many variations on the basic self-training algorithm. Some prior work only trains the student on a small number of unlabeled examples having the highest confidence (Rosenberg et al., 2005; McClosky et al., 2006; Sohn et al., 2020). Recently, other work has opted to train the student on all available data, regardless of teacher confidence (Vu et al., 2021). Additionally, self-training can be performed with either the whole distribution (soft labels) or max probability label (hard labels) (Karamanolakis et al., 2021).

Our results are in Table 4 and suggest that while there is no clear winner between hard and soft labels, training the student on a broad distribution of data is best.

	CDR	SemEval	Youtube	Chemprot
Hard	56.17	65.11	**95.24**	**60.95**
Hard + Thresh	54.34	62.78	93.52	60.65
Soft	**57.92**	**65.56**	94.14	59.34
Soft + Thresh	55.6	62.77	94.09	57.88

Table 4: Variations on the self-training algorithm.

PCA mean subtraction It is common practice to center the data by subtracting the means of each covariate away from a feature matrix prior to computing PCA (Mishra et al., 2017). Our ARI procedure did not employ this mean subtraction trick because intuitively, centering the data prior to PCA prevents the first PCs from being dominated by the variables' means, but this "mean load" is what we intend to capture and control for with our proce-

	TREC	SMS	IMDB
ARI	87.19	95.74	88.33
ARI + mean subtraction	86.71	95.6	87.97

Table 5: PCA mean subtraction ablation (PCA + tree variant).

dure. To validate this hypothesis, we enabled mean subtraction for ARI on 3 datasets. Table 5 gives the results and we can observe a slight decrease in performance, -0.32% on average, when mean subtraction is enabled.

Transferability of rules One possible limitation of ARI is that the rules overfit to their data and are extremely limited to a specific setting. To investigate this, we experimented with swapping the rules between related spam detection datasets SMS and Youtube and found a slight drop in performance (SMS: 95.74 -> 95.26, Youtube: 94.95 -> 93.72). We also tried unrelated datasets IMDB and ChemProt and found that performance drops more (IMDB: 88.98 -> 88.1, ChemProt: 60.1 -> 55.89), but not severely, perhaps because the model learns to ignore most of the rules and defaults to plain self-training (rule attention scores support this: for the matched data, 48.3% of attention went to rules on average while only 21.1% for the unmatched data). We conclude that while the learned rules are adapted to their training data, they are indeed transferable to other domains to some degree, although we note that this work assumes access to in-domain training sets, and that out-of-domain or zero-shot generalization is outside the paper's scope.

4 Interpretability

As discussed in Section 2.2, the behavior of the aggregation layer $a(\cdot)$ can be traced to individual rules, which are themselves human readable and interpretable. This is because the output of $a(\cdot)$ is a linear combination of attention scores and rule predictions (Equation 1). In other words, if the attention score for rule r_j on example x_i is s_i^j, then the strength of rule r_j's contribution to the model's final prediction is exactly s_i^j/Q.

See the Appendix for case studies showing the impact of individual rules on model behavior.

To further demonstrate the system's interpretability, we grouped examples according to their difficulty[4] and measured the cumulative effect of rules on model behavior (i.e., $\sum_j s_i^j/Q$) for each category. The results are given in Table 6. We observe that much of ARI's gains come from the hard-to-predict examples, and that these difficult examples may be associated with increased rule reliance.

	Easy	Medium	Hard
Youtube			
Bert (Acc)	100	95.96	79.36
ARI (Acc)	100 (*45.2*)	94.35 (*48.5*)	87.3 (*49.2*)
ChemProt			
Bert (Acc)	94.27	77.83	43.78
ARI (Acc)	92.78 (*57.4*)	77.97 (*62.9*)	45.02 (*62.4*)

Table 6: Model performance and rule reliance according to whether examples are easy, medium, or hard to predict. The average cumulative effect of the rules $\mathbb{E}_{i \in G}[\ \sum_j s_i^j/Q\ ]$ for each group G is given in parentheses.

5 Related Work

Our research draws on a number of related areas of research, including Neuro-Symbolic computation, semi-supervised learning, and weak supervision.

Neuro-symbolic approaches seek to unite Symbolic AI, which from the 1950's until the mid 1990's was the dominant paradigm of AI research (Crevier, 1993; Russell and Norvig, 2002), with statistical machine learning and neural networks. For example, there is work that uses discrete parses to template neural network components (Arabshahi et al., 2018; Mao et al., 2019; Yi et al., 2018). There is also work that seeks to embed symbolic knowledge into network parameters via special loss functions (Xu et al., 2018; Seo et al., 2021) or carefully curated datasets (Lample and Charton, 2019; Clark et al., 2020; Saeed et al., 2021) and architectures (Trask et al., 2018). Other related work seeks to incorporate logical constraints into text generation models (Wang et al., 2021; Lu et al., 2020).

Our framework is further inspired by semi-supervised learning research that leverages labeled and unsupervised data. Our baseline PET model comes from a family of algorithms that leverage prompting and model ensembling for greater data efficiency (Schick and Schütze, 2020a,b). There is also research on pulling in demonstration examples from the training set (Gao et al., 2020), automatic prompt generation (Zhang et al., 2021b; Li and

[4]Following (Swayamdipta et al., 2020), we used the entropy of BERTs predicted label distribution as a measure of example difficulty. We ranked examples according to this measure, then split them into hard (above the 75th percentile), medium (25-75th percentile) and easy (below 25th percentile).

Liang, 2021), and leveraging extra datasets and tasks for data augmentation when data is scarce (Du et al., 2020; Vu et al., 2021).

Our self-training approach is similar to the knowledge distillation literature (Hinton et al., 2015; Gou et al., 2021) where a "student" model is trained to imitate the predictions of a "teacher" model. In our case, the teacher is not a separate model but a frozen student plus rule aggregation layer.

Another close body of research taps into weak sources of supervision like regular expressions, keywords, and knowledge base alignment (Mintz et al., 2009; Augenstein et al., 2016; Ratner et al., 2017). Researchers have incorporated these weak supervision signals into self-training procedures like ours (Karamanolakis et al., 2021), as well as constructing procedural generators for boosting weak supervision signals (Zhang et al., 2021a) and interactive pipelines for machine-assisted rule construction (Zhang et al., 2022; Galhotra et al., 2021; Maheshwari et al., 2020). There is also research on automatically generating weak labeling functions (Varma and Ré, 2018; Maheshwari et al., 2021) which shares our bag-of-words featurization and regression scoring mechanism.

6 Conclusion

In this paper, we proposed Automatic Rule Induction (ARI), a simple and general-purpose framework for the automatic discovery and integration of symbolic rules into pretrained NLP models. Our results span nine sequence classification and relation extraction tasks and suggest that ARI can improve state-of-the-art algorithms with no manual effort and minimal computational overhead.

Future work could investigate layering ARI beneath other few-shot and semi-supervised algorithms, and improving the underlying rule generation strategies, particularly with causal mechanisms (Feder et al., 2021).

7 Limitations

ARI is not without limitations. We observe that hyperparameter selection is key for quality rule generation (Feurer and Hutter, 2019). Second, as other research has noted (Dodge et al., 2019; Xu et al., 2021), few-shot evaluation protocols remain immature as they rely on small, high variance training sets and static test sets. Last, our procedure works by extrapolating correlations in small training sets, which may result in overfitting and undermine robustness to distribution shift (Sagawa et al., 2020). While the results in Table 2 suggest the end-to-end ARI system is not any more susceptible to spurious correlations than other ML/DL-based methods, i.e. susceptibility is not a big enough issue to prevent SOTA or near-SOTA performance. We hypothesize this may be due to two reasons. First, the rules are heavily regularized: our rule selection model has a strong l2 penalty, our decision trees are generated as part of a stochastic random forest, and the PCA subtraction may also have a regularizing effect. Second, ARI is a hybrid system (neural + rule) which can learn to favor the pre-trained student model when spurious rules fire. Our min-entropy loss function on unlabeled data is designed to encourage such behavior. Concrete examples of spurious rules being ignored can be found in Appendix E.

8 Ethical Considerations

Adding symbolic components to neural systems is a promising way to improve AI trust. Symbolic mechanisms are inherently more interpretable and controllable than black-box function approximators. These components can be reviewed by independent panels and modified to fit the considerations and sensitivities of particular applications.

Microsoft has been 100% carbon neutral since 2012, is committed to being carbon negative by 2030 and removing all of its historical emissions by 2050. This extends to the Microsoft Azure cloud compute engine used for our experiments, which runs on majority renewable energy (clo, 2020).

9 Acknowledgements

We thank Pengcheng He, Giannis Karamanolakis, Hannes Schulz, Yu Shi, Robert Gmyr, Yuwei Fang, Shuohang Wang and many others for their advice.

References

2020. The carbon benefits of cloud computing.

Forough Arabshahi, Sameer Singh, and Animashree Anandkumar. 2018. Combining symbolic expressions and black-box function evaluations in neural programs. *arXiv preprint arXiv:1801.04342*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International conference on learning representations*.

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. *arXiv preprint arXiv:1606.05464*.

Abhijeet Awasthi, Sabyasachi Ghosh, Rasna Goyal, and Sunita Sarawagi. 2020. Learning from rules generalizing labeled exemplars. *arXiv preprint arXiv:2004.06025*.

Benedikt Boecking, Willie Neiswanger, Eric Xing, and Artur Dubrawski. 2020. Interactive weak supervision: Learning useful heuristics for data labeling. *arXiv preprint arXiv:2012.06046*.

Tolga Bolukbasi, Adam Pearce, Ann Yuan, Andy Coenen, Emily Reif, Fernanda Viégas, and Martin Wattenberg. 2021. An interpretability illusion for bert. *arXiv preprint arXiv:2104.07143*.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2020. Transformers as soft reasoners over language. *arXiv preprint arXiv:2002.05867*.

Arman Cohan, Waleed Ammar, Madeleine Van Zuylen, and Field Cady. 2019. Structural scaffolds for citation intent classification in scientific publications. *arXiv preprint arXiv:1904.01608*.

Daniel Crevier. 1993. *AI: the tumultuous history of the search for artificial intelligence*. Basic Books, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A Smith. 2019. Show your work: Improved reporting of experimental results. *arXiv preprint arXiv:1909.03004*.

Jingfei Du, Edouard Grave, Beliz Gunel, Vishrav Chaudhary, Onur Celebi, Michael Auli, Ves Stoyanov, and Alexis Conneau. 2020. Self-training improves pre-training for natural language understanding. *arXiv preprint arXiv:2010.02194*.

Amir Feder, Katherine A Keith, Emaad Manzoor, Reid Pryzant, Dhanya Sridhar, Zach Wood-Doughty, Jacob Eisenstein, Justin Grimmer, Roi Reichart, Margaret E Roberts, et al. 2021. Causal inference in natural language processing: Estimation, prediction, interpretation and beyond. *arXiv preprint arXiv:2109.00725*.

Matthias Feurer and Frank Hutter. 2019. Hyperparameter optimization. In *Automated machine learning*, pages 3–33. Springer, Cham.

Andrea Galassi, Kristian Kersting, Marco Lippi, Xiaoting Shao, and Paolo Torroni. 2020. Neural-symbolic argumentation mining: An argument in favor of deep learning and reasoning. *Frontiers in big Data*, 2:52.

Sainyam Galhotra, Behzad Golshan, and Wang-Chiew Tan. 2021. Adaptive rule discovery for labeling text data. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2217–2225.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819.

Yves Grandvalet and Yoshua Bengio. 2004. Semi-supervised learning by entropy minimization. *Advances in neural information processing systems*, 17.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7).

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*.

Giannis Karamanolakis, Subhabrata Mukherjee, Guoqing Zheng, and Ahmed Hassan Awadallah. 2021. Self-training with weak supervision. *arXiv preprint arXiv:2104.05514*.

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

Guillaume Lample and François Charton. 2019. Deep learning for symbolic mathematics. *arXiv preprint arXiv:1912.01412*.

Dong-Hyun Lee et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Yinghao Li, Pranav Shetty, Lucas Liu, Chao Zhang, and Le Song. 2021. Bertifying the hidden markov model for multi-source weakly supervised named entity recognition. *arXiv preprint arXiv:2105.12848*.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *arXiv preprint arXiv:2205.05638*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Ximing Lu, Peter West, Rowan Zellers, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Neurologic decoding:(un) supervised neural text generation with predicate logic constraints. *arXiv preprint arXiv:2010.12884*.

Ayush Maheshwari, Oishik Chatterjee, Krishnateja Killamsetty, Ganesh Ramakrishnan, and Rishabh Iyer. 2020. Semi-supervised data programming with subset selection. *arXiv preprint arXiv:2008.09887*.

Ayush Maheshwari, Krishnateja Killamsetty, Ganesh Ramakrishnan, Rishabh Iyer, Marina Danilevsky, and Lucian Popa. 2021. Learning to robustly aggregate labeling functions for semi-supervised data programming. *arXiv preprint arXiv:2109.11410*.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. 2019. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2014. Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv preprint arXiv:1412.6632*.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159.

Dheeraj Mekala and Jingbo Shang. 2020. Contextualized weak supervision for text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 323–333.

Mike Mintz, Steven Bills, Rion Snow, and Dan 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.

Sidharth Prasad Mishra, Uttam Sarkar, Subhash Taraphder, Sanjay Datta, D Swain, Reshma Saikhom, Sasmita Panda, and Menalsh Laishram. 2017. Multivariate statistical data analysis-principal component analysis (pca). *International Journal of Livestock Research*, 7(5):60–78.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993.

Kamal Nigam and Rayid Ghani. 2000. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93.

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.

Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, page 269. NIH Public Access.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. 2005. Semi-supervised self-training of object detection models.

Stuart Russell and Peter Norvig. 2002. Artificial intelligence: a modern approach.

Mohammed Saeed, Naser Ahmadi, Preslav Nakov, and Paolo Papotti. 2021. Rulebert: Teaching soft rules to pre-trained language models. *arXiv preprint arXiv:2109.13006*.

Esteban Safranchik, Shiying Luo, and Stephen Bach. 2020. Weakly supervised sequence tagging from noisy rules. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5570–5578.

Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. 2020. An investigation of why overparameterization exacerbates spurious correlations. In *International Conference on Machine Learning*, pages 8346–8356. PMLR.

Timo Schick and Hinrich Schütze. 2020a. Exploiting cloze questions for few shot text classification and natural language inference. *arXiv preprint arXiv:2001.07676*.

Timo Schick and Hinrich Schütze. 2020b. It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*.

Sungyong Seo, Sercan O Arik, Jinsung Yoon, Xiang Zhang, Kihyuk Sohn, and Tomas Pfister. 2021. Controlling neural networks with rule representations. *arXiv preprint arXiv:2106.07804*.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*.

Kihyuk Sohn, David Berthelot, Chun-Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. 2020. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *arXiv preprint arXiv:2001.07685*.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. *arXiv preprint arXiv:2004.09297*.

Swabha Swayamdipta, Roy Schwartz, Nicholas Lourie, Yizhong Wang, Hannaneh Hajishirzi, Noah A Smith, and Yejin Choi. 2020. Dataset cartography: Mapping and diagnosing datasets with training dynamics. *arXiv preprint arXiv:2009.10795*.

Shavrina Tatiana and Malykh Valentin. 2021. How not to lie with a benchmark: Rearranging nlp leaderboards. *arXiv preprint arXiv:2112.01342*.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Thomas P Trappenberg. 2019. Machine learning with sklearn. In *Fundamentals of Machine Learning*, pages 38–65. Oxford University Press.

Andrew Trask, Felix Hill, Scott E Reed, Jack Rae, Chris Dyer, and Phil Blunsom. 2018. Neural arithmetic logic units. *Advances in neural information processing systems*, 31.

Paroma Varma and Christopher Ré. 2018. Snuba: Automating weak supervision to label training data. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 12, page 223. NIH Public Access.

Tu Vu, Minh-Thang Luong, Quoc V Le, Grady Simon, and Mohit Iyyer. 2021. Strata: Self-training with task augmentation for better few-shot learning. *arXiv preprint arXiv:2109.06270*.

Yufei Wang, Can Xu, Huang Hu, Chongyang Tao, Stephen Wan, Mark Dras, Mark Johnson, and Daxin Jiang. 2021. Neural rule-execution tracking machine for transformer-based text generation. *Advances in Neural Information Processing Systems*, 34.

Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not explanation. *arXiv preprint arXiv:1908.04626*.

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Broeck. 2018. A semantic loss function for deep learning with symbolic knowledge. In *International conference on machine learning*, pages 5502–5511. PMLR.

Liang Xu, Xiaojing Lu, Chenyang Yuan, Xuanwei Zhang, Huilin Xu, Hu Yuan, Guoao Wei, Xiang Pan, Xin Tian, Libo Qin, et al. 2021. Fewclue: A chinese few-shot learning evaluation benchmark. *arXiv preprint arXiv:2107.07498*.

Ziyi Yang, Yinfei Yang, Daniel Cer, and Eric Darve. 2021. A simple and effective method to eliminate the self language bias in multilingual representations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5825–5832.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B Tenenbaum. 2018. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *arXiv preprint arXiv:1810.02338*.

Jieyu Zhang, Yue Yu, Yinghao Li, Yujing Wang, Yaming Yang, Mao Yang, and Alexander Ratner. 2021a. Wrench: A comprehensive benchmark for weak supervision. *arXiv preprint arXiv:2109.11377*.

Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. 2021b. Differentiable prompt makes pre-trained language models better few-shot learners. *arXiv preprint arXiv:2108.13161*.

Rongzhi Zhang, Yue Yu, Pranav Shetty, Le Song, and Chao Zhang. 2022. Prboost: Prompt-based rule discovery and boosting for interactive weakly-supervised learning. *arXiv preprint arXiv:2203.09735*.

A Appendix A: Reproducibility

To construct our ngram feature matrices, we built a vocabulary of size 1600 using NLTK's WordNet lemmatizer and `word_tokenize` tokenizer. We used the built-in English stopwords list, as well as a max document frequency cutoff of 0.95 and minimum token frequency cutoff of 4, and ngrams up to length 2.

Hyperparameters are given below. For each algorithm we describe the search space and say in parentheses which settings had the best validation performance for each dataset (and thus were selected for testing). Unless otherwise stated, we used a learning rate of 1e-5 for all algorithms, a batch size of 24, max sequence length of 128, and optimized using Adam (Kingma and Ba, 2014). Note that we used the originally published hyperparameters for the Min Entropy, VAT, and MoE baselines.

BERT:

- No search.

LMFT:

- Pretraining epochs: 1, 3 (all datasets), 5.

Self-Train:

- Number of self-training iterations: 15, 25 (all datasets), 40.
- Ratio of labeled-to-unlabeled train steps: 0.7 (all datasets), 1.0.

PET:

- Learning rate 1e-6, 1e-5 (all datasets), 1e-4.
- Ensemble model train epochs: 2 (AGNews, IMDB), 3 (CDR, ChemProt, SciCite, TREC), 5 (SemEval, SMS, Youtube).
- Final classifier train epochs: 2, 3 (everything else), 5 (SemEval, Youtube), 10.
- Our prompting templates are given in Table 7.

ARI:

- Rule embedding size: 100.
- Number of rules: 16 (AGNews, CDR, ChemProt, SMS, Youtube), 32 (IMDB, SciCite, SemEval, TREC), 64.
- Inference with student (AGNews, CDR, IMDB, SciCite), teacher (ChemProt, SemEval, SMS, Youtube, TREC).
- Tree rule threshold: 0.95 (SciCite), 0.8 (all other datasets).
- Number of self-training iterations: 15, 25 (all datasets), 40.
- Ratio of labeled-to-unlabeled train steps: 0.7 (all datasets), 1.0.
- Filter selection: described in Section .

B Appendix B: Example Rules

We provide some concrete examples of unigram rules generated by ARI on the SMS dataset. This dataset involves detecting whether text messages are spam or not, so they are relatively easy to reason about in an intuitive sense.

Ngram linear rules:

```
if "..." in x predict HAM else ABSTAIN
if ": )" in x predict HAM else ABSTAIN
if ".." in x predict HAM else ABSTAIN
if "txt" in x predict SPAM else ABSTAIN
if "service" in x predict SPAM else ABSTAIN
if "." in x predict HAM else ABSTAIN
if "claim" in x predict SPAM else ABSTAIN
if "dating" in x predict SPAM else ABSTAIN
if "?" in x predict HAM else ABSTAIN
if "ringtone" in x predict SPAM else ABSTAIN
if "ok" in x predict HAM else ABSTAIN
if "reply" in x predict SPAM else ABSTAIN
if "say" in x predict HAM else ABSTAIN
if "free" in x predict SPAM else ABSTAIN
if "home" in x predict HAM else ABSTAIN
if "fancy" in x predict SPAM else ABSTAIN
```

Some of these rules make sense; text messages asking recipients to "claim" items that are "free" or "fancy" are probably spam. Smiley faces (":)") and proper punctuation (".", "?") are normal things to write in a text message.

Ngram tree rules:

Note that our random forest was implemented with the sklearn package (Trappenberg, 2019) and so we use the same display format as their `sklearn.tree.export_text.` function: each node evaluates the frequency of it's associated string and branches accordingly.

```
|--- mob week <= 0.50
|   |--- ? ? 1000 <= 0.50
|   |   |--- awarded ? <= 0.50
|   |   |   |--- class: HAM
|   |   |--- awarded ? >  0.50
|   |   |   |--- class: SPAM
|   |--- ? ? 1000 >  0.50
|   |   |--- class: SPAM
|--- mob week >  0.50
|   |--- class: SPAM
```

```
|--- ? 500 <= 0.50
|   |--- won ? <= 0.50
|   |   |--- c <= 0.50
|   |   |   |--- class: HAM
|   |   |--- c >  0.50
|   |   |   |--- class: SPAM
|   |--- won ? >  0.50
|   |   |--- class: SPAM
|--- ? 500 >  0.50
|   |--- class: SPAM
```

```
|--- urgent ! <= 0.50
```

	Label words	Template 1	Template 2
SMS	normal, junk	[mask] : [text]	[mask] message: [text]
AGNews	world, sports, business, tech	[mask]: [text]	[mask] news: [text]
IMDB	bad, good	[mask]: [text]	[mask] review: [text]
SemEval	cause, component, content, destination, origin instrument, member, message, product	[mask]: [text]	[mask] text: [text]
CDR	no, yes	[mask]: [text]	[mask] text: [text]
SciCite	background, method, result	[mask]: [text]	[mask] text: [text]
ChemProt	part, regulator, up, down, agony, antagonist modify, together, product, not	[mask]: [text]	[mask] Passage: [text]
Youtube	normal, junk	[mask]: [text]	[mask] comment: [text]
TREC	description, entity, human, abbreviation location, number	[mask]: [text]	[mask] statement: [text]

Table 7: PET verbalization templates and label word mappings.

```
|   |--- ringtone <= 0.50
|   |   |--- send stop <= 0.50
|   |   |   |--- class: HAM
|   |   |--- send stop >  0.50
|   |   |   |--- class: SPAM
|   |--- ringtone >  0.50
|   |   |--- class: SPAM
|--- urgent ! >  0.50
|   |--- class: SPAM


|--- ! <= 0.50
|   |--- 750 <= 0.50
|   |   |--- win <= 0.50
|   |   |   |--- class: HAM
|   |   |--- win >  0.50
|   |   |   |--- class: SPAM
|   |--- 750 >  0.50
|   |   |--- class: SPAM
|--- ! >  0.50
|   |--- . <= 0.50
|   |   |--- cash <= 0.50
|   |   |   |--- class: HAM
|   |   |--- cash >  0.50
|   |   |   |--- class: SPAM
|   |--- . >  0.50
|   |   |--- line <= 0.50
|   |   |   |--- class: HAM
|   |   |--- line >  0.50
|   |   |   |--- class: SPAM
```

We find that these rules are less readily interpretable than directly using ngrams, but generally make sense. For example, the second to last rule suggests that if a text message contains an exclamation mark and large number (750) followed by "win", the message is spam ("win 750$!") but without the word "win" the message is probably not spam, (there are plenty of non-spammy reasons to talk about large numbers in a text message).

PCA tree rules: These rules, being constructed from a dense feature space, are less readily interpretable. We denote each feature dimension by the ngram it originated from, wrapped in quotes and followed by (+PCA).

```
|--- come (+PCA) <= -0.03
|   |--- '& free' (+PCA) <= -0.00
|   |   |--- class: HAM
|   |--- '& free' (+PCA) >  -0.00
|   |   |--- 'won' (+PCA) <= -0.02
|   |   |   |--- class: HAM
|   |   |--- 'won' (+PCA) >  -0.02
|   |   |   |--- class: SPAM
|--- 'come' (+PCA) >  -0.03
|   |--- 'ringtone' (+PCA) <= 0.50
|   |   |--- 'latest' (+PCA) <= 0.50
|   |   |   |--- class: HAM
|   |   |--- 'latest' (+PCA) >  0.50
|   |   |   |--- class: SPAM
|   |--- 'ringtone' (+PCA) >  0.50
|   |   |--- class: SPAM


|--- ''m' (+PCA) <= -0.05
|   |--- 'lt ; #' (+PCA) <= -0.13
|   |   |--- 'win' (+PCA) <= -0.02
|   |   |   |--- class: HAM
|   |   |--- 'win' (+PCA) >  -0.02
|   |   |   |--- class: SPAM
|   |--- 'lt ; #' (+PCA) >  -0.13
|   |   |--- 'win ? ?' (+PCA) <= -0.01
|   |   |   |--- class: HAM
|   |   |--- 'win ? ?' (+PCA) >  -0.01
|   |   |   |--- class: SPAM
|--- ''m' (+PCA) >  -0.05
|   |--- 'r' (+PCA) <= -0.02
|   |   |--- 't &' (+PCA) <= -0.01
|   |   |   |--- class: HAM
|   |   |--- 't &' (+PCA) >  -0.01
|   |   |   |--- class: SPAM
|   |--- 'r' (+PCA) >  -0.02
|   |   |--- 'free' (+PCA) <= 0.49
|   |   |   |--- class: HAM
|   |   |--- 'free' (+PCA) >  0.49
|   |   |   |--- class: SPAM


|--- 'd' (+PCA) <= -0.02
```

```
|   |--- 'un-redeemed' (+PCA) <= -0.00
|   |   |--- class: HAM
|   |--- 'un-redeemed' (+PCA) >  -0.00
|   |   |--- ', love' (+PCA) <= -0.01
|   |   |   |--- class: SPAM
|   |   |--- ', love' (+PCA) >  -0.01
|   |   |   |--- class: HAM
|--- 'd' (+PCA) >  -0.02
|   |--- 'video' (+PCA) <= 0.50
|   |   |--- 'stop' (+PCA) <= 0.96
|   |   |   |--- class: HAM
|   |   |--- 'stop' (+PCA) >  0.96
|   |   |   |--- class: SPAM
|   |--- 'video' (+PCA) >  0.50
|   |   |--- class: SPAM
```

C Appendix C: Teacher and Student Performance

As described in Section 3.3, one can use either the teacher or student model for ARI inference. Table 8 has the results and suggests that their performance is similar, and that there is no clear winner.

	SemEval	SMS	AGNews	CDR
Teacher	65.56	95.74	91.22	57.10
Student	65.47	95.39	91.37	57.92

Table 8: Relative performance of the teacher and student model, using the same filter settings as in Table 2.

D Appendix D: Rule Performance

Table 9 gives the performance of the rules by themselves, using the best combination of filters for downstream performance (described in Section 3.3). Interestingly, we find that the rules do not always outperform BERT, even on the small number of examples they fire on. We hypothesize that the contextualized nature of the teacher's embedding mechanism may be helping it further determine when rules should be applied.

	SemEval		AGNews	
	Cov.	Pre.	Cov.	Pre.
BERT	1.0	0.75	1.0	90.69
Ngram + linear	0.10	47.80	0.03	96.84
Ngram + tree	0.12	59.25	0.05	94.12
PCA + tree	0.16	50.93	0.06	94.74

Table 9: Performance of BERT and the rules themselves, given as F1 score on the examples where a rule fired. We also provide the coverage, i.e. the proportion of test examples where rules were firing.

E Appendix E: Samples

We provide some examples of unigram-based ARI and BERT outputs on the SemEval dataset below. For ease of understanding, we only select examples where only a small number of ngram rules fired.

1. **TEXT**: A hinge assembly attaches a cover pivotally to a base of an electronic device and has a pivoting leaf and a stationary leaf . Entity 1: assembly, entity 2: cover.

   BERT: Instrument-Agency

   ARI: Component-Whole

   LABEL: Component-Whole

   Attn:

   0.60

   ```
   if 'has' in x predict Cause-Effect else ABSTAIN
   ```

   0.97

   ```
   if 'has' in x predict Component-Whole else ABSTAIN
   ```

   Interestingly, in this case the same token was mapped to two rules, and the system learned to dynamically prefer one over the other based on context.

2. **TEXT**: She left the engine running because the car was full of snakes used in her exotic routine . Entity 1: snakes, entity 2: car.

   BERT: Member-Collection

   ARI: Content-Container

   LABEL: Content-Container

   Attn:

   0.52

   ```
   if 'wa' in x predict CONTENT-CONTAINER else ABSTAIN
   ```

   This is an example of a rule helping the model correctly fix its prediction. "wa" often maps to "was" with our tokenizer. This rule and the above "has" rule are both words that convey a sense of two properties or entities being related to one another, which intuitively seem related to solving the SemEval task (relation classification).

3. **TEXT**: I still shiver as I remember trying to page through economics texts by the flicker from candles while clad in overcoat , scarf , and little knitted gloves with the fingertips cut off , in the 4 p.m . Entity 1: candles, entity 2: flicker.

BERT: Member-Collection

ARI: Cause-Effect

LABEL: Cause-Effect

Attn:

0.78

```
if ',' in x predict MEMBER-COLLECTION else ABSTAIN
```

Interestingly, in this case the rule was incorrect and had high attention but the teacher model (correctly) favored of the student's prediction. Note also that this is a pretty bad rule, as it is a general and nonspecific punctuation marker.

4. **TEXT**: Hands wield the sword in the realm of the flesh , but the intellect wields the pen in the realm of understanding , or of the spirit . Entity 1: pen, entity 2: intellect.

   BERT: Instrument-Agency

   ARI: Member-Collection

   LABEL: Instrument-Agency

   Attn:

   0.99

```
if ',' in x predict MEMBER-COLLECTION else ABSTAIN
```

This is an example of the same spurious rule as before likely causing the ARI system to make an error.

	AGNews	CDR	ChemProt	IMDB	SciCite	SemEval	SMS	TREC	Youtube	Avg.
ARI + BERT	90.87	57.92	60.01	88.33	83.76	65.1	95.74	87.19	94.95	79.05
ARI + DeBERTaV3	91.38	60.09	60.48	87.19	81.97	66.41	97.20	86.28	94.88	79.40
PET + BERT	91.46	51.16	53.9	88.49	75.13	64.45	95.68	83.42	95.36	75.76
PET + DeBERTaV3	91.14	52.63	54.84	88.92	75.97	63.50	96.14	84.90	95.09	76.29

Table 10: Comparison between PET and ARI (PCA+tree variant) with BERT and DeBERTaV3 backbones. Both methods improved slightly (PET by 0.65% and ARI by 0.11% on average) but ARI remains better overall.

	AGNews	CDR	ChemProt	IMDB	SciCite	SemEval	SMS	TREC	Youtube	Avg.
BERT	90.33	69.56	64.12	86.84	79.84	82.23	98.6	86.8	91.66	82.66
SELF	89.52	70.65	65.34	86.92	80.21	85.43	98.6	87.6	93.33	83.55
VAT	89.51	72.28	67.98	89.51	80.28	87.64	99.2	87.4	94.16	84.77
PET	90.53	69.48	65.84	88.71	78.12	87.5	96.73	86.8	96.83	83.81
PR-BOOST	88.9		67.1							
ARI (ours)	89.76	72.06	68.71	89.68	81.67	87.08	99	87.4	97.5	85.30
T-few (3B)	94.2	72	70.98	94.46	82.9	86.9	99.15	95	95	87.2

Table 11: Accuracies for the main results (Table 2).