WALNUT: A Benchmark on Semi-weakly Supervised Learning for Natural Language Understanding

Guoqing Zheng
Microsoft Research
zheng@microsoft.com

Giannis Karamanolakis
Columbia University
gkaraman@cs.columbia.edu

Kai Shu
Illinois Institute of Technology
kshu@iit.edu

Ahmed Hassan Awadallah
Microsoft Research
hassanam@microsoft.com

Abstract

Building machine learning models for natural language understanding (NLU) tasks relies heavily on labeled data. Weak supervision has been proven valuable when large amount of labeled data is unavailable or expensive to obtain. Existing works studying weak supervision for NLU either mostly focus on a specific task or simulate weak supervision signals from ground-truth labels. It is thus hard to compare different approaches and evaluate the benefit of weak supervision without access to a unified and systematic benchmark with diverse tasks and real-world weak labeling rules. In this paper, we propose such a benchmark, named WALNUT[1], to advocate and facilitate research on weak supervision for NLU. WALNUT consists of NLU tasks with different types, including document-level and token-level prediction tasks. WALNUT is the first semi-weakly supervised learning benchmark for NLU, where each task contains weak labels generated by multiple real-world weak sources, together with a small set of clean labels. We conduct baseline evaluations on WALNUT to systematically evaluate the effectiveness of various weak supervision methods and model architectures. Our results demonstrate the benefit of weak supervision for low-resource NLU tasks and highlight interesting patterns across tasks. We expect WALNUT to stimulate further research on methodologies to leverage weak supervision more effectively. The benchmark and code for baselines are available at `aka.ms/walnut_benchmark`.

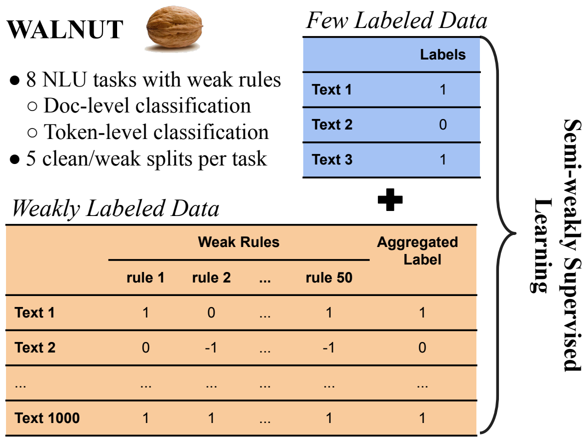


Figure 1: WALNUT, a benchmark with 8 NLU tasks with real-world weak labeling rules. Each task in WALNUT comes with few labeled data and weakly labeled data for semi-weakly supervised learning.

1 Introduction

To tackle natural language understanding (NLU) tasks via supervised learning, high-quality labeled examples are crucial. Recent advances on large pre-trained language models (Peters et al., 2018; Devlin et al., 2018; Radford et al., 2019) lead to impressive gains on NLU benchmarks, including GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019), at the assumption that large amount of labeled examples are available. For many real-world applications, however, it is expensive and time-consuming to manually obtain large-scale high-quality labels, while it is relatively easier to obtain auxiliary supervision signals, or *weak supervision*, as a viable source to boost model performance without expensive data annotation process.

Learning from weak supervision for NLU tasks is attracting increasing attention. Various types of weak supervision have been considered, such as knowledge bases (Mintz et al., 2009; Xu et al., 2013), keywords (Karamanolakis et al., 2019; Ren et al., 2020), regular expression patterns (Augenstein et al., 2016), and other metadata such as user interactions in social media (Shu et al., 2017). Also, inspired by recent advances from semi-supervised learning, *semi-weakly supervised learning* methods which leverage both a small set of clean labels and a larger set of weak supervision (Papandreou et al., 2015; Hendrycks et al., 2018; Shu et al., 2019; Mazzetto et al., 2021; Karamanolakis

[1] WALNUT: Semi-**WeA**kly supervised **L**earning for **N**atural language **U**nderstanding **T**estbed

et al., 2021; Maheshwari et al., 2021; Zheng et al., 2021) are emerging to further boost task performance. However, a unified and systematic evaluation benchmark supporting *both weakly and semi-weakly* supervised learning for NLU tasks is rather limited. On the one hand, many existing works only study specific NLU tasks with weak supervision, thus evaluations of proposed techniques leveraging weak supervision on a small set of tasks do not necessarily generalized onto other NLU tasks. On the other hand, some works rely on simulated weak supervision, such as weak labels corrupted from ground-truth labels (Hendrycks et al., 2018), while real-world weak supervision signals can be far more complex than simulated ones. Furthermore, existing weakly and semi-weakly supervised approaches are evaluated on different data with different metrics and weak supervision sources, making it difficult to understand and compare.

To better advocate and facilitate research on leveraging weak supervision for NLU, in this paper we propose WALNUT (Figure 1), a semi-weakly supervised learning benchmark of NLU tasks with real-world weak supervision signals. Following the tradition of existing benchmarks (e.g., GLUE), we propose to cover different types of NLU tasks and domains, including document-level classification tasks (e.g., sentiment analysis on online reviews, fake news detection on news articles), and token-level classification tasks (e.g., named entity recognition in news and biomedical documents). WALNUT provides few labeled and many weakly labeled examples (Figure 1) and encourages a consistent and robust evaluation of different techniques, as we will describe in Section 3.

In addition to the proposed benchmark, in Section 4.2 we shed light on the benefit of weak supervision for NLU tasks in a collective manner, by evaluating several representative weak and semi-weak supervision methods for and several base models of various sizes (e.g., BiLSTM, BERT, RoBERTa), leading to more than 2,000 groups of experiments. Our large-scale analysis demonstrates that weak supervision is valuable for low-resource NLU tasks and that there is large room for performance improvement, thus motivating future research. Also, by computing the average performance across tasks and model architectures, we show surprising new findings. First, simple techniques for aggregating multiple weak labels (such as unweighted majority voting) achieve better performance than more complex weak supervision paradigms. Second, weak supervision has smaller benefit in larger base models such as RoBERTa, because larger pre-trained models can already achieve impressively high performance using just a few clean labeled data and no weakly labeled data at all. We identify several more challenges on leveraging weak supervision for NLU tasks and shed light on possible future work based on WALNUT.

The main contributions of this paper are: (1) We propose a new benchmark on semi-weakly supervised learning for NLU, which covers eight established annotated datasets and various text genres, dataset sizes, and degrees of task difficulty; (2) We conduct an exploratory analysis from different perspectives to demonstrate and analyze the results for several major existing weak supervision approaches across tasks; and (3) We discuss the benefits and provide insights for potential weak supervision studies for representative NLU tasks.

2 Related Work

2.1 Weak Supervision for NLU

Document-level classification Existing works on weakly supervised learning for document-level classification attempt to correct the weak labels by incorporating a loss correction mechanism for text classification (Sukhbaatar et al., 2014; Patrini et al., 2017). Other works further assume access to a small set of clean labeled examples (Hendrycks et al., 2018; Ren et al., 2018; Varma and Ré, 2018; Shu et al., 2020b). Recent works also consider the scenario where weak signals are available from multiple sources (Ratner et al., 2017; Meng et al., 2018; Ren et al., 2020) to exploit the redundancy as well as the consistency in the labeling information. Despite the recent progress on weak supervision for text classification, there is no agreed upon benchmark that can guide future directions and development of NLU tasks in semi-weakly supervised setting.

Token-level classification Weak supervision has also been studied for token-level classification (sequence tagging) tasks, focusing on Named Entity Recognition (NER). One of the most common approaches is distant supervision (Mintz et al., 2009), which uses knowledge bases to heuristically annotate training data. Besides distant supervision, several weak supervision approaches have recently addressed NER by introducing various types of labeling rules, for example based on keywords, lexicons, and regular expressions (Fries et al., 2017; Ratner et al., 2017; Shang et al., 2018; Safranchik et al., 2020; Lison et al., 2020; Li et al., 2021). WALNUT integrates existing weak rules into a unified representation and evaluation format.

2.2 NLU Benchmarks

Accompanying the emerging of large pre-trained language models, NLU benchmarks has been a focus for NLP research, including GLUE (Wang et al., 2018) and SuperGLUE (Wang et al., 2019). On such benchmarks, the major focus is put on obtaining best possible performance (He et al., 2020) under the full training setting, which assumes that a large quantity of manually labeled examples are available for all tasks. Few-shot NLU benchmarks exist (Schick and Schütze, 2021; Xu et al., 2021; Ye et al., 2021; Mukherjee et al., 2021), however these do not contain weak supervision. Though research in weak supervision in NLU has gained significant interest (Hendrycks et al., 2018; Shu et al., 2019; Zheng et al., 2021), most of these work either focus on a small set of tasks or simulate weak supervision signals from ground-truth labels, hindering its generalization ability to real-world NLU tasks. The lack of a unified test bed covering different NLU task types and data domains motivates us to construct such a benchmark to better understand and leverage semi-weakly supervised learning for NLU in this paper.

Different from existing work based on crowdsourcing (Hovy et al., 2013; Gokhale et al., 2014) to obtain noisy labels, we focus specifically on the *semi-weakly supervised* learning setting, where we collect tasks with weak labels obtained from human-written labeling rules. (Zhang et al., 2021) is concurrent work that also features weak supervision for various (not necessarily text-based) tasks and assumes a purely weakly supervised setting, i.e., no clean labeled data is available. In contrast, WALNUT focuses on NLU tasks under a more-realistic semi-weakly supervised setting and, as we show in Section 3, a small amount of clean labeled data plays an important role in determining the benefit of weak supervision for a target task.

3 WALNUT

3.1 Benchmark Construction Principles

We first describe the design principles guiding the benchmark construction.

Task Selection Criterion We aim to create a testbed which covers a broad range of NLU tasks where *real-world weak supervision signals are available*. To this end, WALNUT includes eight English text understanding tasks from diverse domains, ranging from news articles, movie reviews, merchandise reviews, biomedical corpus, wikipedia documents, to tweets. The eight tasks are categorized evenly into two types, namely document classification and token classification (sequence labeling). It's worth noting that *we didn't create any labeling rules ourselves to avoid bias, but rather opted with labeling rules which already exist and are extensively studied by previous research.* Therefore, WALNUT does not include other NLU tasks, such as natural language inference and question answering, as we are not aware of previous research with human labeling rules for these tasks.

Semi-weakly Supervised Learning Setting While many previous works studied weak supervision in a purely weakly supervised setting, recent advances in few-shot and semi-supervised learning suggest that a small set of cleanly labeled examples together with unlabeled examples greatly helps boosting the task performance. Though large scale labeled examples for a task is difficult to collect, we acknowledge that it's rather practical to collect a small set of labeled examples. In addition, recent methods leveraging weak supervision also demonstrate greater gains of combining a small set of labeled examples with large weakly labeled examples (Hendrycks et al., 2018; Shu et al., 2019; Zheng et al., 2021). Therefore, WALNUT is designed to emphasize the semi-weakly supervised learning setting. Specifically, each dataset contains both a small number of clean labeled instances and a large number of weakly-labeled instances. Each weakly-labeled instance comes with multiple weak labels (assigned by multiple rules) and a single aggregated weak label derived from weak rules. *Note that this way WALNUT can be naturally used to support the conventional weakly supervised setting by ignoring the provided clean labels.*

Consistent and Robust Evaluation To address discrepancies in evaluation protocols from existing research on weak supervision and to better account for the small set of clean examples per task, WALNUT is constructed to promote systematic and robust evaluations across all eight tasks. Specif-

Table 1: Statistics of the eight document- and token-level tasks in WALNUT. See Section 3.2 for details.

Dataset	AGNews	IMDB	Yelp	GossipCop	CoNLL	NCBI	WikiGold	LaptopReview
Label granularity	doc.	doc.	doc.	doc.	token	token	token	token
Task	topic	sentiment	sentiment	fake	NER	NER	NER	NER
Domain	news	movies	restaurants	news	news	biomed	web	tech
# Classes	4	2	2	2	9	3	9	3
# Train-clean ($\|D_C\|$)	80	40	40	40	180	60	360	150
# Train-weak ($\|D_W\|$)	4,439	16,626	10,954	6,462	13,861	532	995	2,286
# Dev	12,000	2,500	3,800	1,430	3,250	99	169	609
# Test	12,000	2,500	3,800	957	3,453	99	170	800
# Weak rules	9	8	8	3	50	12	16	12

ically, for each task, we first determine the number of clean examples to sample with pilot experiments (with the rest treated as weakly labeled examples by applying the corresponding weak labeling rules), such that the weakly supervised examples can be still helpful with the small clean examples present (typically 20-50 per class; see Sec. 3.3 for details); second, to consider sampling uncertainty, we repeat the sampling process for the desired number of clean examples 5 times and provide all 5 splits in WALNUT. Methods on WALNUT are expected to be using all 5 pre-computed splits and reporting the mean and variance of its performance.

To summarize, WALNUT can facilitate research on weakly- and semi-weakly supervised learning by offering the following:

- Eight NLU tasks from diverse domains;
- For each task, five pairs of clean and weakly labeled samples for robust evaluation;
- For each individual weakly labeled example, all weak labels from multiple rules and a single aggregated weak label.

3.2 Task Categories

Here, we describe the eight tasks in WALNUT (Table 1), grouped into four document-level classification tasks (Section 3.2.1) and four token-level classification tasks (Section 3.2.2).

3.2.1 Document-level Classification

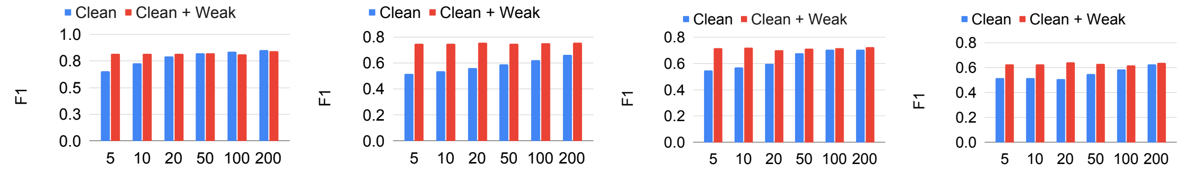
The goal of document-level classification tasks is to classify a sequence of tokens $x_1, \ldots, x_N$ to a class $c \in C$, where C is a pre-defined set of classes. We consider binary and multi-class classification problems from different application domains such as sentiment classification (Zhang et al., 2015), fake news detection (Shu et al., 2020c), and topic classification (Zhang et al., 2015). Concretely, we include the following widely-used document-level text classification datasets: AGNews (Zhang et al., 2015), Yelp (Zhang et al., 2015), IMDB (Maas et al., 2011) and GossipCop (Shu et al., 2020a).

For Yelp, IMDB, and AGNews, the weak rules are derived from the text using keyword-based heuristics, third-party tools as detailed in (Ren et al., 2020). For GossipCop, the weak labeling rules are derived from social context information accompanying the news articles, including related users' social engagements on the news items (e.g., user comments in Twitter). For example, a weak labeling rule for fake news can be "If a news piece has a standard deviation of user sentiment scores greater than a threshold, then the news is weakly labeled as fake news. " (Shu et al., 2020c).

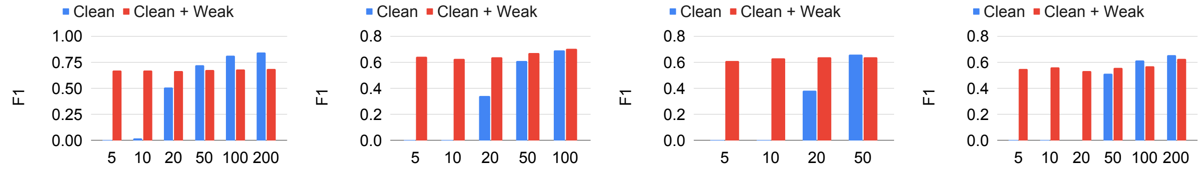
3.2.2 Token-level Classification

The goal of token-level classification tasks is to classify a sequence of tokens $x_1, \ldots, x_N$ to a sequence of tags $y_1, \ldots, y_N \in C'$, where C' is a pre-defined set of tag classes (e.g., person or organization). As one of the most common token-level classification tasks, Named Entity Recognition (NER) deals with recognizing categories of named entities (e.g., person, organization, location) and is important in several NLP pipelines, including information extraction and question answering.

We include in WALNUT the following four NER datasets from different domains, for which weak rules are available: CoNLL (Sang and De Meulder, 2003), the NCBI Disease corpus (Doğan et al., 2014), WikiGold (Balasuriya et al., 2009) and the LaptopReview corpus (Pontiki et al., 2016) from the SemEval 2014 Challenge. For the CoNLL and WikiGold dataset, we use weak rules provided by (Lison et al., 2020). For the NCBI and LaptopReview dataset, we use weak rules provided by (Safranchik et al., 2020).



(a) AG News (b) IMDB (c) Yelp (d) Gossipcop



(e) CoNLL (f) NCBI (g) WikiGold (h) LaptopReview

Figure 2: F1 score by varying (in the x-axis) the number of clean instances per class considered in the clean training set (D_C). The importance of weak supervision is more evident for settings with smaller numbers of instances, where the gap in performance between the "Clean" approach and "Clean+Weak" approach is larger. For a robust evaluation across tasks, WALNUT provides five clean/weak splits per task. See Section 3.3 for details.

3.3 Dataset Pre-Processing

To construct a semi-weakly supervised learning setting, we split the training dataset for each task into a small subset with clean labels (D_C) and a large subset with weak labels (D_W). For robust evaluation, we create five different clean/weak train splits as we noticed that the model performances may vary with different clean train instances. The validation/test sets are always the same across splits.

Because of different dataset characteristics (e.g., differences in number of classes, difficulty), we choose the size for D_C per dataset via pilot studies. (After having selected the instances for the D_C, we consider the remaining instances as part of the D_W split.) We defined the size of D_C such that we demonstrate the benefits of weak supervision and at the same time leave substantial room for improvement in future research. To this end, we compare the performances of the same base classification model (e.g., BiLSTM), trained using only D_C ("Clean" approach) v.s. using both D_C and D_W ("Clean+Weak" approach). As shown in Figure 2, for each dataset, we choose a small size of D_C, such that the "Clean+Weak" approach has a substantially higher F1 score than the "Clean" approach and at the same time the "Clean" approach has no trivial F1 score.

The statistics of the pre-processed datasets included in WALNUT are shown in Table 1.

4 Baseline Evaluation in WALNUT

In this section, we describe the baselines and evaluation procedure (Section 4.1), and discuss evaluation results in WALNUT (Section 4.2). Our results highlight the value of weak supervision, important differences across different baselines, and the potential utility of WALNUT for future research on weak supervision.

4.1 Baselines and Evaluation Procedure

We evaluate several baseline approaches in WALNUT by considering different base models (text encoders) and different (semi-)weakly supervised learning methods to train the base model.

Encoder Models To encode input text, we experiment with various text encoders, ranging from shallow LSTMs to large pre-trained transformer-based encoders (Vaswani et al., 2017). In particular, we consider a series of models with increasing model size: Bi-directional LSTM with Glove embeddings (Pennington et al., 2014), DistilBERT (Sanh et al., 2019), BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), BERT-large, and RoBERTa-large. For each text encoder, a classification head is placed on top of the encoder to perform the task. For more details on the base model configurations see Appendix A.1.

Learning Methods Given the semi-weakly supervised setting in WALNUT, we evaluate eight supervision approaches in the following categories:

- *Learning from clean labeled examples only*. The model is trained on only the small amount of available clean examples D_C, a naive baseline method leveraging no weak supervision, which we denote as **C**.

- *Learning from weakly labeled examples only*. The model is trained on all weakly labeled examples D_W. To produce a single weak label from the multiple labeling rules for training, we aggregate the rules via two methods: majority voting (denoted by **W**) and Snorkel (Ratner et al., 2017) (denoted by **Snorkel**).

- *Learning from both clean and weakly labeled examples*. The model is trained with both D_C and D_W in a weakly-supervised setting. The first two baselines in this category is simply concatenating D_C and the aggregated weak labels (from either **W** and **Snorkel**), and the model is trained on the combination. We denote these two as **C+W** and **C+Snorkel**, respectively. We also test three recent semi-weakly supervised learning methods which proposed better ways to leverage both D_C and D_W: **GLC** which is a loss correction approach (Hendrycks et al., 2018), **MetaWN** which is a meta-learning approach to learn the importance of weakly labeled examples (Shu et al., 2019; Ren et al., 2018) and **MLC**, a meta-learning approach to learn to correct the weak labels (Zheng et al., 2021).

To establish an estimate of the ceiling performance on WALNUT, for each task we also train with all clean training examples in the original dataset (denoted by **Full Clean**).

Experimental Procedure For a robust evaluation, we repeat each experiment five times on the five splits of D_C and D_W (clean and weak examples for each task; see Section 3.3), and report the average scores and the standard deviation across the five runs. In WALNUT, we report the average micro-average F1 score on the test set.[2] Datasets and code for WALNUT are publicly available at `aka.ms/walnut_benchmark`.

4.2 Experimental Results and Analysis

Table 2 shows the main evaluation results on WALNUT. Rows correspond to supervision methods for the base model, columns correspond to tasks, and each block corresponds to a different base model. Unless explicitly mentioned, in the rest of this section we will compare approaches based on their average performance across tasks (rightmost column in Table 2).

As expected, training with Full Clean achieves the highest F1 score, corresponding to the high-resource setting where all clean labeled data are available. Such method is not directly comparable to the rest of the methods but serves as an estimate of the ceiling performance for WALNUT. Training with only limited clean examples achieves the lowest overall F1 score: in the low-resource setting, which is the main focus in WALNUT, using just the available clean subset (D_C) is not effective.

Weak supervision is valuable for low-resource NLU. "W" and "Snorkel" achieve better F1 scores than "C" for many base models: even using only weakly-labeled data in D_W is more effective than using just D_C, thus demonstrating that simple weak supervision approaches can be useful in the low-resource setting. Approaches such as "C+W" and "C+Snorkel" lead to further improvements compared to "C" and "Snorkel", thus highlighting that even simple approaches for integrating clean and weak labeled data (here by concatenating D_C and D_W) are more effective than considering each separately.

There is no clear winner in WALNUT. Our results in Table 4.2 indicate that the performance of weak supervision techniques varies substantially across tasks. Therefore, it is important to evaluate such techniques in a diverse set of tasks to achieve a fair comparison and more complete picture of their performance. The performance of various techniques also varies across different splits (See Table 14 in Appendix for variances of all experiments). Interestingly, "C+W" and "C+Snorkel" sometimes perform better than more complicated approaches, such as GLC, MetaWN and MLC.

Larger base models achieve better overall performance. We further aggregate statistics across tasks, methods, and base models in Table 3. The bottom row reports the average performance across methods for each base model and leads to a consistent ranking in F1 score among base models: BiLSTM $\leq$ DistilBERT $\leq$ BERT-base $\leq$ RoBERTa-base. Observing higher scores for larger transformer models such as RoBERTa agrees with previous observations (Brown et al., 2020). Interestingly, switching from BERT-base to BERT-large (and from RoBERTa-base to RoBERTa-large) in base model architecture leads to marginal improvement, suggesting the need to explore more effective learning methods leveraging weak supervision.

[2] For token-level F1, we use the conlleval implementation: `https://huggingface.co/metrics/seqeval`

Table 2: Main results on WALNUT with F1 score (in %) on all tasks. The rightmost column reports the average F1 score across all tasks. (MLC is not shown for BERT-large and RoBERTa-large due to OOM.)

Method	AGNews	IMDB	Yelp	GossipCop	CoNLL	NCBI	WikiGold	LaptopReview	AVG
				BiLSTM (20M parameters)					
Full Clean	89.4	83.1	86.4	64.5	31.9	69.9	21.8	62.6	63.7
C	79.5	56.2	59.5	50.8	00.8	58.2	**15.8**	42.3	45.4
W	78.0	75.2	70.8	62.0	11.1	52.3	02.7	49.4	50.2
Snorkel	79.9	75.4	**76.0**	61.4	06.7	52.5	02.7	49.4	52.1
C+W	82.0	**75.6**	70.2	**64.1**	**17.2**	56.8	15.7	51.2	**52.5**
C+Snorkel	**82.9**	75.4	66.5	62.6	07.7	**59.2**	10.7	53.8	52.4
GLC	56.5	72.2	63.7	60.5	05.1	58.9	08.7	**55.2**	47.6
MetaWN	55.2	72.7	65.5	58.2	00.0	53.9	03.4	51.6	45.1
MLC	55.3	72.3	65.7	52.5	00.0	52.5	05.9	51.5	43.7
				DistilBERT-base (66M parameters)					
Full Clean	92.1	88.8	93.7	75.1	88.6	75.7	79.7	75.8	83.7
C	80.8	71.2	73.1	55.3	51.4	57.7	69.5	53.0	64.0
W	72.2	75.0	70.2	70.8	66.9	62.0	57.4	53.8	66.0
Snorkel	70.2	70.7	65.9	68.4	64.3	62.9	56.3	54.0	65.1
C+W	83.3	74.8	71.5	**71.4**	66.9	66.2	64.0	57.3	68.5
C+Snorkel	**84.3**	**81.7**	**81.8**	69.1	64.6	67.8	64.4	57.5	**71.4**
GLC	67.8	74.1	68.1	67.3	**72.4**	**72.8**	**71.7**	**66.8**	70.1
MetaWN	70.0	74.4	69.3	70.0	65.7	64.2	58.5	58.2	66.3
MLC	70.4	74.3	69.4	69.6	69.2	66.2	58.3	58.0	66.9
				BERT-base (110M parameters)					
Full Clean	92.5	90.0	74.7	74.7	89.4	78.4	81.1	76.2	82.1
C	82.9	63.8	60.3	57.1	67.3	66.6	**71.9**	54.6	65.6
W	72.3	75.5	69.6	69.0	67.5	59.5	56.7	55.9	65.8
Snorkel	73.7	72.9	65.6	68.2	65.1	60.9	53.8	56.2	66.0
C+W	**80.1**	81.8	71.3	68.4	68.4	67.9	65.0	59.2	68.9
C+Snorkel	76.2	**82.6**	**75.3**	67.1	65.9	69.9	64.3	59.6	70.1
GLC	68.8	75.7	68.8	68.1	**74.7**	**74.7**	70.7	**65.8**	**70.9**
MetaWN	72.8	75.2	68.1	69.8	66.9	66.7	58.9	59.2	67.2
MLC	73.0	74.7	70.0	**71.3**	70.4	68.4	58.5	59.7	68.2
				RoBERTa-base (125M parameters)					
Full Clean	92.8	92.4	95.9	77.2	91.2	83.1	87.2	80.2	87.5
C	**84.1**	74.5	70.2	57.4	72.9	72.9	78.2	61.3	71.4
W	66.4	76.1	**70.4**	71.4	64.9	69.9	64.1	58.9	67.8
Snorkel	71.9	70.1	66.3	69.2	61.2	70.0	61.8	59.7	67.5
C+W	70.6	**76.5**	**70.4**	72.2	64.1	74.0	71.6	61.2	68.9
C+Snorkel	74.6	68.2	66.4	71.4	62.2	73.4	72.2	61.6	68.8
GLC	67.6	74.9	69.0	68.0	**74.6**	**79.1**	**79.6**	**71.5**	**73.0**
MetaWN	69.6	75.4	69.0	71.8	63.8	69.9	63.5	62.5	68.2
MLC	70.4	74.5	69.9	**72.9**	68.3	74.3	63.1	63.6	69.6
				BERT-large (336M parameters)					
Full Clean	92.5	91.4	94.9	73.5	90.2	80.5	82.8	78.9	85.6
C	72.5	65.4	68.4	57.8	67.2	69.7	**73.9**	51.1	65.8
W	68.5	**75.9**	70.7	69.3	65.7	62.0	57.1	54.2	65.4
Snorkel	73.3	70.9	65.8	70.0	63.6	67.3	57.2	54.4	66.5
C+W	73.4	74.8	**71.8**	70.2	66.7	70.7	66.9	55.7	67.6
C+Snorkel	**73.6**	71.3	65.9	**71.3**	63.6	69.7	63.4	57.2	67.0
GLC	67.1	74.6	67.3	69.8	**71.8**	**76.1**	68.1	**65.4**	**70.0**
MetaWN	71.6	74.2	67.0	70.8	64.4	70.1	53.9	45.9	64.7
				RoBERTa-large (355M parameters)					
Full Clean	93.1	94.4	96.9	78.5	91.3	83.5	87.7	80.4	88.2
C	**86.1**	69.1	**84.8**	69.1	76.4	77.7	77.1	60.6	**75.1**
W	74.3	**77.7**	70.5	73.2	63.6	67.4	61.2	57.3	68.2
Snorkel	75.5	72.6	67.1	69.3	61.1	68.1	61.0	59.2	67.9
C+W	71.9	77.4	70.6	71.6	**63.8**	71.2	70.4	60.2	68.5
C+Snorkel	74.0	69.0	66.5	**73.7**	61.1	71.8	69.1	62.4	68.5
GLC	67.8	75.8	68.7	64.1	56.7	**80.0**	**78.3**	**68.4**	70.0
MetaWN	68.6	66.2	71.1	64.6	63.2	69.5	59.3	61.5	65.5
				Rules (no base model)					
Rules	61.8	73.9	65.9	73.5	61.3	64.7	52.2	60.0	64.1

Table 3: Average F1 score across the eight tasks in WALNUT. The bottom row computes the average F1 score across tasks and supervision methods. The three right-most columns report the average F1 score across model architectures and all tasks ("All"), document-level tasks ("Doc"), and token-level tasks ("Token").

	Base Model Architecture						Average Results		
Method	BiLSTM	DistilBERT	BERT	RoBERTa	BERT-large	RoBERTa-large	All	Doc	Token
Full Clean	63.7	83.7	82.1	87.5	85.6	88.2	81.8	86.6	77.0
C	45.4	64.0	65.6	71.4	65.8	**75.1**	64.5	68.7	60.3
W	50.2	66.0	65.8	67.8	65.4	68.2	63.9	71.9	55.9
Snorkel	52.1	65.1	66.0	67.5	66.5	67.9	64.2	70.4	57.9
C+W	**52.5**	68.5	68.9	68.9	67.6	68.5	65.8	**73.6**	58.0
C+Snorkel	52.4	**71.4**	70.1	68.8	67.0	68.5	66.3	73.0	59.7
GLC	47.6	70.1	**70.9**	**73.0**	**70.0**	70.0	**66.9**	68.6	**65.3**
MetaWN	45.1	66.3	67.2	68.2	64.7	65.5	62.8	69.2	56.4
AVG	51.1	69.4	69.6	71.6	69.1	71.5			

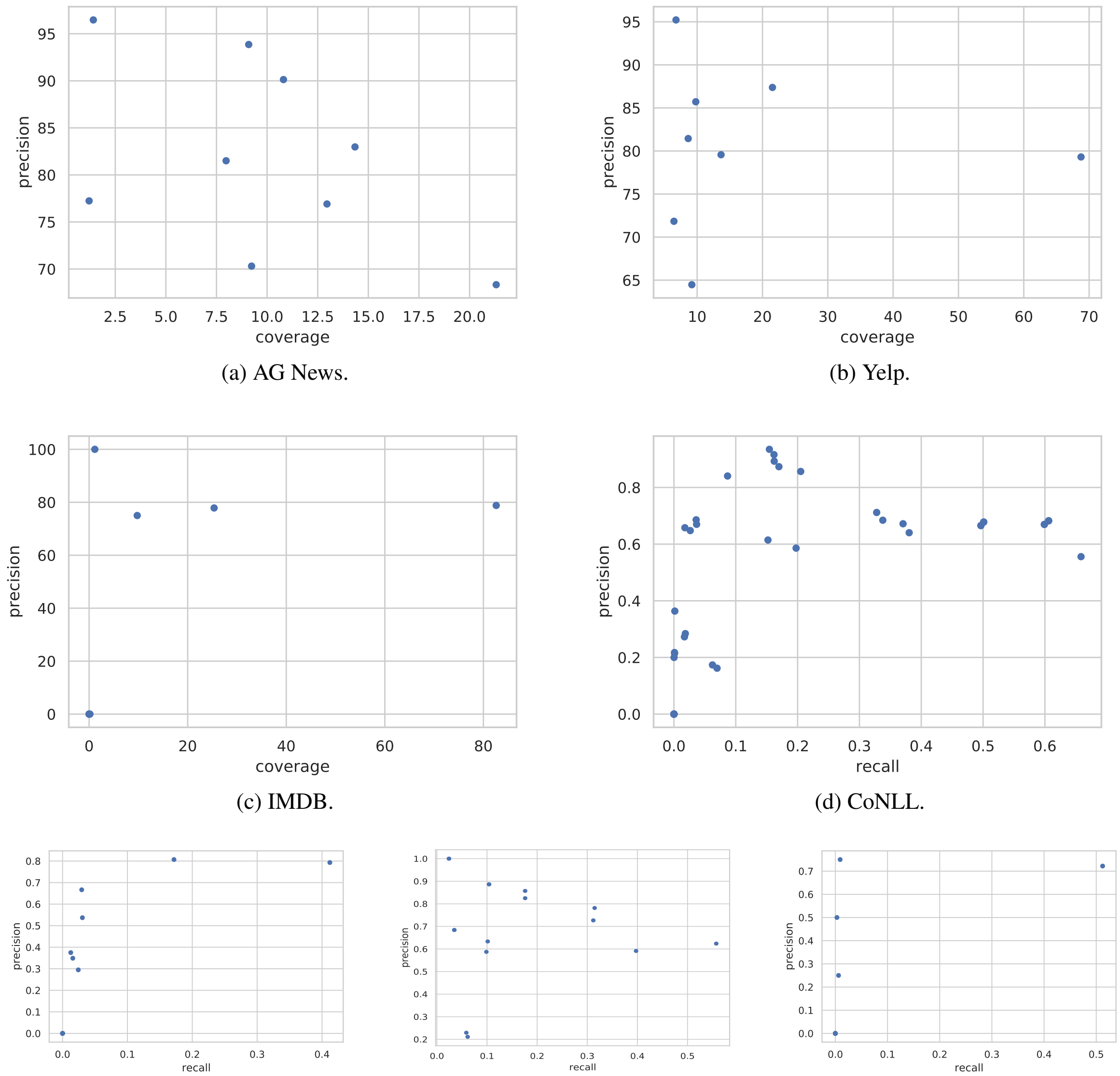
Table 4: Overall performance gain and gap of all weak supervision methods (Weak Sup, by averaging performance of W, Snorkel, C+W, C+Snorkel, GLC, MetaWN and MLC) against no weak supervision (C) and full clean training. Note that RoBERTa-large in included here, as the standard deviation of its performance with different splits on tasks varies significantly (See Table 14 in Appendix) hence using its performance mean as an indicator is less conclusive.

	BiLSTM	DistilBERT	BERT	RoBERTa	BERT-large	AVG
Perf. gain: Weak Sup − C	3.90	4.21	2.98	-2.07	1.32	2.07
Perf. gap: Full Clean − Weak Sup	14.42	15.47	13.58	18.13	18.51	16.00

Weak supervision has smaller benefit in larger base models. Another question that we attempt to address in WALNUT is on whether weak supervision equally benefits each base model architecture. To quantify such benefit we compare the performance differences between models trained using semi-weak supervision and models trained using clean data only. The "Weak Sup" approach in Table 4 is computed as the average F1 score across all semi-weak supervision methods (C+W, C+Snorkel, GLC, and MetaWN). The performance gap between "Weak Sup" and "C" (training with few clean data only) is smaller for larger models. Additionally, the performance gap between "Full Clean" (full clean data training) and "Weak Sup" approach is larger for larger models. The two above observations highlight that weak supervision has smaller benefit in larger models. An important future research direction is to develop better learning algorithms and improving the effectiveness of weak supervision in larger models.

Analysis of weak rules. For now, we have focused on the evaluation of base models trained using weak labels generated by multiple weak labeling rules. It is interesting also to decouple the base model performance from the rule aggregation technique (e.g., majority voting, Snorkel) that was used to generate the training labels, which is an essential modeling component for weak supervision. The bottom row in Table 2 ("Rules") reports the test performance of rules computed by taking the majority voting of weak labels on the test instances. (For test instances that are not covered by any rules, a random class is predicted.) Such majority label is available in our pre-processed datasets. Interestingly, "Rules" sometimes outperforms base models trained using weak labels ("W", "Snorkel"). Note however that "Rules" assumes *access to all weak labels* on the test set, which might not always be available. On the other hand, the base model learns text features beyond the heuristic-based rules and does not require access to rules during test time and thus can be applied for any test instance.

For a more in-depth analysis of the rule quality, WALNUT also supports the analysis of individual rules and multi-source aggregation techniques, such as majority voting or Snorkel. Figure 3 shows a precision-recall scatter plot for each rule on each of the dataset in WALNUT. For instance, in the CoNLL dataset rules vary in characteristics, where most rules have a relatively low recall while there are a few rules that have substantially higher recall than the rest. Across datasets, we observe that rules



(a) AG News. (b) Yelp. (c) IMDB. (d) CoNLL. (e) NCBI. (f) WikiGold. (g) LaptopReview.

Figure 3: Scatterplots of precision-recall for weak supervision rules. Each point corresponds to a rule. (GossipCop is omitted as it contains only three rules.)

have higher precision than recall, as most rules are sparse, i.e., apply to a small number of instances in the dataset (e.g., instances containing a specific keyword). Similar trends are observed on other datasets as well. For detailed descriptions of all weak rules in all datasets, refer to Table 6 - 13 in the appendix.

5 Conclusions

Motivated by the lack of a unified evaluation platform for semi-weakly supervised learning for low-resource NLU, in this paper we propose a new benchmark WALNUT covering a broad range of data domains to advocate research on leveraging both weak supervision and few-shot clean supervision. We evaluate a series of different semi-weakly supervised learning methods with different model architecture on both document-level and token-level classification tasks, and demonstrate the utility of weak supervision in real-world NLU tasks. We find that no single semi-weakly supervised learning method wins on WALNUT and there is still gap between semi-weakly supervised learning and fully supervised training. We expect WALNUT to enable systematic evaluations of semi-weakly supervised learning methods and stimulate further research in directions such as more effective learning paradigms leveraging weak supervision.

Acknowledgements

We would like to thank Yichuan Li for helping adapt several weak supervision baselines for document-level classification tasks, and the reviewers for constructive feedback.

References

Isabelle Augenstein, Tim Rocktäschel, Andreas Vlachos, and Kalina Bontcheva. 2016. Stance detection with bidirectional conditional encoding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 876–885.

Dominic Balasuriya, Nicky Ringland, Joel Nothman, Tara Murphy, and James R Curran. 2009. Named entity recognition in wikipedia. In *Proceedings of the 2009 Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources (People's Web)*, pages 10–18.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

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

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.

Jason Fries, Sen Wu, Alex Ratner, and Christopher Ré. 2017. Swellshark: A generative model for biomedical named entity recognition without labeled data. *arXiv preprint arXiv:1704.06360*.

Chaitanya Gokhale, Sanjib Das, AnHai Doan, Jeffrey F Naughton, Narasimhan Rampalli, Jude Shavlik, and Xiaojin Zhu. 2014. Corleone: Hands-off crowdsourcing for entity matching. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 601–612.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention.

Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. 2018. Using trusted data to train deep networks on labels corrupted by severe noise. In *NeurIPS*.

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

Giannis Karamanolakis, Daniel Hsu, and Luis Gravano. 2019. Leveraging just a few keywords for fine-grained aspect detection through weakly supervised co-training. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Giannis Karamanolakis, Subhabrata Mukherjee, Guoqing Zheng, and Ahmed Hassan. 2021. Self-training with weak supervision. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 845–863.

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. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*.

Pierre Lison, Jeremy Barnes, Aliaksandr Hubin, and Samia Touileb. 2020. Named entity recognition without labelled data: A weak supervision approach. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1518–1533.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Andrew 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.

Ayush Maheshwari, Oishik Chatterjee, Krishnateja Killamsetty, Ganesh Ramakrishnan, and Rishabh Iyer. 2021. Semi-supervised data programming with subset selection. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*.

Alessio Mazzetto, Cyrus Cousins, Dylan Sam, Stephen H Bach, and Eli Upfal. 2021. Adversarial multi class learning under weak supervision with performance guarantees. In *Proceedings of the 38th International Conference on Machine Learning*.

Yu Meng, Jiaming Shen, Chao Zhang, and Jiawei Han. 2018. Weakly-supervised neural text classification. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 983–992.

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.

Subhabrata Mukherjee, Xiaodong Liu, Guoqing Zheng, Saghar Hosseini, Hao Cheng, Greg Yang, Christopher Meek, Ahmed Hassan Awadallah, and Jianfeng Gao. 2021. Few-shot learning evaluation in natural language understanding. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

George Papandreou, Liang-Chieh Chen, Kevin P Murphy, and Alan L Yuille. 2015. Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1742–1750.

Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. 2017. Making deep neural networks robust to label noise: A loss correction approach. In *CVPR*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237.

Maria Pontiki, Dimitrios Galanis, Haris Papageorgiou, Ion Androutsopoulos, Suresh Manandhar, Mohammad Al-Smadi, Mahmoud Al-Ayyoub, Yanyan Zhao, Bing Qin, Orphée De Clercq, et al. 2016. Semeval-2016 task 5: Aspect based sentiment analysis. In *International workshop on semantic evaluation*, pages 19–30.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment*, 11(3):269–282.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to reweight examples for robust deep learning. *arXiv preprint arXiv:1803.09050*.

Wendi Ren, Yinghao Li, Hanting Su, David Kartchner, Cassie Mitchell, and Chao Zhang. 2020. Denoising multi-source weak supervision for neural text classification. *arXiv preprint arXiv:2010.04582*.

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.

Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

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

Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Jingbo Shang, Jialu Liu, Meng Jiang, Xiang Ren, Clare R Voss, and Jiawei Han. 2018. Automated phrase mining from massive text corpora. *IEEE Transactions on Knowledge and Data Engineering*, 30(10):1825–1837.

Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. 2019. Meta-weight-net: Learning an explicit mapping for sample weighting. In *Advances in Neural Information Processing Systems*, pages 1917–1928.

Kai Shu, Deepak Mahudeswaran, Suhang Wang, Dongwon Lee, and Huan Liu. 2020a. Fakenewsnet: A data repository with news content, social context, and spatiotemporal information for studying fake news on social media. *Big Data*, 8(3):171–188.

Kai Shu, Subhabrata Mukherjee, Guoqing Zheng, Ahmed Hassan Awadallah, Milad Shokouhi, and Susan Dumais. 2020b. Learning with weak supervision for email intent detection. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1051–1060.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. 2017. Fake news detection on social media: A data mining perspective. *ACM SIGKDD explorations newsletter*, 19(1):22–36.

Kai Shu, Guoqing Zheng, Yichuan Li, Subhabrata Mukherjee, Ahmed Hassan Awadallah, Scott Ruston, and Huan Liu. 2020c. Leveraging multi-source weak social supervision for early detection of fake news. *arXiv preprint arXiv:2004.01732*.

Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. 2014. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*.

Paroma Varma and Christopher Ré. 2018. Snuba: automating weak supervision to label training data. *VLDB*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Superglue: a stickier benchmark for general-purpose language understanding systems. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 3266–3280.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Liang Xu, Xiaojing Lu, Chenyang Yuan, Xuanwei Zhang, Hu Yuan, Huilin Xu, Guoao Wei, Xiang Pan, and Hai Hu. 2021. Fewclue: A chinese few-shot learning evaluation benchmark. *CoRR*, abs/2107.07498.

Wei Xu, Raphael Hoffmann, Le Zhao, and Ralph Grishman. 2013. Filling knowledge base gaps for distant supervision of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Qinyuan Ye, Bill Yuchen Lin, and Xiang Ren. 2021. Crossfit: A few-shot learning challenge for cross-task generalization in nlp.

Jieyu Zhang, Yue Yu, Yinghao Li, Yujing Wang, Yaming Yang, Mao Yang, and Alexander Ratner. 2021. Wrench: A comprehensive benchmark for weak supervision. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Guoqing Zheng, Ahmed Hassan Awadallah, and Susan Dumais. 2021. Meta label correction for noisy label learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 35.

A Appendix

Limitations and Broader Impact

The proposed benchmark is likely to stimulate research on weakly supervised learning for NLU, and offer the research community on weakly supervised learning a unified testbed for evaluating new methodologies developed for low-resource NLU. For many practical NLU applications, large amount of manually labeled data is unavailable or expensive to obtain due to either cost or privacy concerns, resorting to proxy signals such as weak supervision is a viable solution to mitigate this annotation scarce problem. We hope WALNUT would provide such an evaluation environment to advocate progress in this direction.

Limitations. Due to the lack of existing real-world weak supervision for many NLU tasks, WALNUT does not include NLU tasks such as Natural Language Inference for which it is hard to construct weak supervision rules. Also, currently WALNUT only considers English data; a possible extension is to also include multi-lingual corpus with weak supervision available to boost the performance of multi-lingual language models with weakly supervised learning.

A.1 Implementation Details

We implement all baseline experiments with PyTorch and each experiment runs on a single NVIDIA GPU. Below are hyper-parameter specifications for all baseline methods (hyperparameters not mentioned below are given default values):

- Full Clean, C, C+W, Snorkel, C+Snorkel: Batch size is 32 for document-level classification datasets and 16 for the token-level classification datasets. The code for Snorkel is adapted from: `https://github.com/snorkel-team/snorkel`. Each training experiment is conducted for the 10 epochs with the checkpoint with the best validation performance saved for evaluation on the test set.

- GLC: Code is adapted from `https://github.com/mmazeika/glc`. Batch size is 16 for the 4 document-level classification datasets and 8 for the 4 token-level classification datasets. Each experiment trains for 10 epochs with the checkpoint with the best validation performance saved for evaluation on test set.

- MetaWN: Code is adapted from `https://github.com/xjtushujun/meta-weight-net`. Batch size is 8 for the 4 document-level classification datasets and 4 for the 4 token-level classification datasets. The meta-network is a three-layer feed-forward network with hidden dimension of 128. Each experiment trains for 10 epochs with the checkpoint with the best validation performance saved for evaluation on test set.

- MLC: Code is adapted from `https://github.com/microsoft/MLC`. Batch size is 8 for the 4 document-level classification datasets and 4 for the 4 token-level classification datasets. The meta-network is a three-layer feed-forward network with hidden dimension of 128; the label embedding dimension used in the meta-network is 64. Each experiment trains for 10 epochs with the checkpoint with the best validation performance saved for evaluation on test set.

To encode input text, we experiment with various text encoders, ranging from shallow LSTMs to large pre-trained transformerbased encoders (Vaswani et al., 2017):

- BiLSTM-based encoder: the BiLSTM implementations are all based on 50-dimensional pre-trained glove word embeddings (Pennington et al., 2014) and bi-directional LSTMs with hidden size 128. Note that our implementation is different than other BiLSTM implementations used by previous work, which are based on 100-dimensional word embeddings and LSTM hidden size 300. This renders our BiLSTM models roughly 3 times smaller than those used by previous work, thus the numbers are not directly comparable. We chose a smaller model capacity for BiLSTMs to contrast the performance with larger models including DistilBERT and others to show the importance of model capacity on WALNUT. During training, we use a learning rate of 0.005 for all BiLSTM-based models.

- Transformer-based encoders: we consider pre-trained DistilBERT (Sanh et al., 2019), BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), BERT-large, and RoBERTa-large.

We fine-tune these models (via the hugging-face library) using task-specific classification heads on top of the encoder and a learning rate of 0.00001.

B Additional Benchmark Details

B.1 Document-level classification

- **AGNews:** and multi-class topic classification (world vs. sports vs. business vs. sci/tech) on news articles from the AGNews dataset (Zhang et al., 2015).
- **Yelp:** binary sentiment classification (negative vs. positive) of Yelp restaurant reviews (Zhang et al., 2015).
- **IMDB:** binary sentiment classification (negative vs. positive) of IMDB movie reviews (Maas et al., 2011).
- **GossipCop:** binary fake news detection (fake vs. not fake) on news articles from the GossipCop[3] fact-checking websites. The GossipCop dataset is part of the fake news detection benchmark FakeNewsNet (Shu et al., 2020a). (We only include the results of Gossipcop to represent fake news classification task as the results for Politifact are similar.)

B.2 Token-level classification

According to the BIO tagging scheme, "B," "I," and "O," represent the beginning, inside, and outside, of a named entity span, respectively. (Not extracting any values corresponds to a sequence of "O"-only tags.) Consider, for example, named entity recognition in the CoNLL dataset:

Tokens:	Barack	Obama	lives	in	Washington
Tags:	B-PER	I-PER	O	O	B-LOC

- **CoNLL:** the CoNLL 2003 dataset (Sang and De Meulder, 2003) contains news articles from Reuters (split into sentences). In total, there are 35,089 entities from 4 types: organization (ORG), person (PER), location (LOC), and miscellaneous (MISC). Tag classes C': ['O', 'B-PER', 'I-PER', 'B-ORG', 'I-ORG', 'B-LOC', 'I-LOC', 'B-MISC', 'I-MISC']
- **NCBI:** the NCBI Disease corpus (Doğan et al., 2014) contains PubMed abstracts with 6,866 disease mentions. Tag types: ['O', 'B', 'I']
- **WikiGold:** the WikiGold dataset (Balasuriya et al., 2009) contains English Wikipedia articles that were randomly selected and manually annotated with the same entity types as CoNLL. Tag classes C': ['O', 'B-PER', 'I-PER', 'B-ORG', 'I-ORG', 'B-LOC', 'I-LOC', 'B-MISC', 'I-MISC']
- **LaptopReview:** the Laptop Review corpus from the SemEval 2014 Challenge (Pontiki et al., 2016) contains 3,012 mentions to laptop features. Tag types C': ['O', 'B', 'I']

Table 5 shows detailed statistics for token-level classification datasets. More dataset statistics are provided in Table 1. Tables 6-13 show detailed information for all rules. Figures 4-10 show examples of weak rules for various datasets.

[3] https://www.gossipcop.com/

Table 5: Extra token-level statistics for the token-level classification datasets.

	CoNLL	NCBI	WikiGold	LaptopReview
# train tokens	203,621	135,572	31,560	41,525
# dev tokens	51,362	23,789	3,683	9,970
# test tokens	46,435	24,219	3,762	11,884

Table 6: List of rules for the AGNews dataset. The rules are the same as the tagging rules in (Zhang et al., 2015). The Python implementations can be found in: https://github.com/weakrules/Denoise-multi-weak-sources/blob/master/rules-noisy-labels/Agnews/angews_rule.py

Rule name	Description
1. world1	Keyword-based detection of the WORLD topic
2. world2	Keyword-based detection of the WORLD topic
3. sports1	Keyword-based detection of the SPORTS topic
4. sports2	Keyword-based detection of the SPORTS topic
5. sports3	Keyword-based detection of the SPORTS topic
6. tech1	Keyword-based detection of the TECH topic
7. tech2	Keyword-based detection of the TECH topic
8. business1	Keyword-based detection of the BUSINESS topic
9. business2	Keyword-based detection of the BUSINESS topic

Table 7: List of rules for the IMDB dataset. The rules are the same as in (Zhang et al., 2015). The Python implementations can be found in: https://github.com/weakrules/Denoise-multi-weak-sources/blob/master/rules-noisy-labels/IMDB/imdb_rule.py

Rule name	Description
1. expression_nexttime	Regex-based detection of POSITIVE sentiment (re-watching expressions)
2. expression_recommend	Regex-based detection of POSITIVE sentiment (recommendation expressions)
3. expression_value	Regex-based detection of POSITIVE sentiment (value expressions)
4. keyword_compare	Keyword-based detection of NEGATIVE sentiment based on movie comparisons
5. keyword_general	Keyword-based detection of POSITIVE and NEGATIVE sentiment
6. keyword_actor	Keyword-based detection of POSITIVE sentiment regarding the actors
7. keyword_finish	Keyword-based detection of NEGATIVE sentiment
8. keyword_plot	Keyword-based detection of POSITIVE and NEGATIVE sentiment regarding the plot

Table 8: List of rules for the Yelp dataset. The rules are the same as in (Zhang et al., 2015). The Python implementations can be found in: https://github.com/weakrules/Denoise-multi-weak-sources/blob/master/rules-noisy-labels/Yelp/yelp_rules.py

Rule name	Description
1. textblob_lf	Model-based detection of POSITIVE and NEGATIVE sentiment (TextBlob model)
2. keyword_recommend	Regex-based detection of POSITIVE sentiment (recommendation expressions)
3. keyword_general	Regex-based detection of POSITIVE and POSITIVE sentiment (general expressions)
4. keyword_mood	Keyword-based detection of POSITIVE and NEGATIVE sentiment based on the user's mood
5. keyword_service	Keyword-based detection of POSITIVE and NEGATIVE sentiment relevant to the service
6. keyword_price	Keyword-based detection of POSITIVE and NEGATIVE sentiment regarding the prices
7. keyword_environment	Keyword-based detection of POSITIVE and NEGATIVE sentiment relevant to the ambience
8. keyword_food	Keyword-based detection of POSITIVE and NEGATIVE sentiment relevant to the food

Table 9: List of rules for the GossipCop dataset. The rules are the same as in (Shu et al., 2020a) (page 8 in http://www.cs.iit.edu/~kshu/files/ecml_pkdd_mwss.pdf).

Rule name	Description
1. mean_scores	User interaction-based detection of FAKE news: If a news piece has a standard deviation of user sentiment scores greater than a threshold τ_1, then the news is weakly labeled as FAKE news.
2. std_scores	User interaction-based detection of FAKE news: If the mean value of users' absolute bias scores - sharing a piece of news – is greater than a threshold τ_2, then the news piece is weakly-labeled as FAKE news.
3. credibility_score	User interaction-based detection of FAKE news: If a news piece has an average credibility score less than a threshold τ_3, then the news is weakly-labeled as FAKE news.

Table 10: List of rules for the CoNLL dataset. The Python implementation of CoNLL rules is provided in the "skweak" repo: https://github.com/NorskRegnesentral/skweak/blob/670fcdec680930ce3e497886d06d61e6a1f2c195/examples/ner/conll2003_ner.py

Rule name	Description
1. date_detector	Heuristic detection of entities of type DATE
2. time_detector	Heuristic detection of entities of type TIME
3. money_detector	Heuristic detection of entities of type MONEY
4. proper_detector	Heuristic detection of proper names based on casing
5. infrequent_proper_detector	Heuristic detection of proper names based on casing + including at least one infrequent token
6. proper2_detector	Heuristic detection of proper names based on casing
7. infrequent_proper2_detector	Heuristic detection of proper names based on casing + including at least one infrequent token
8. nnp_detector	Heuristic detection of sequences of tokens with NNP as POS-tag
9. infrequent_nnp_detector	Heuristic detection of sequences of tokens with NNP as POS-tag + including at least one infrequent token (rank > 15000 in vocabulary)
10. compound_detector	Heuristic detection of proper noun phrases with compound dependency relations
11. infrequent_compound_detector	Heuristic detection of proper noun phrases with compound dependency relations + including at least one infrequent token
12. misc_detector	Heuristic detection of entities of type NORP, LANGUAGE, FAC or EVENT
13. legal_detector	Heuristic detection of entities of type LAW
14. company_type_detector	Gazetteer using a large list of company names
15. full_name_detector	Heuristic function to detect full person names
16. number_detector	Heuristic detection of entities CARDINAL,ORDINAL, PERCENT and QUANTITY
17. snips	Probabilistic parser specialised in the recognition of dates, + times, money amounts, percents, and cardinal/ordinal values
18. core_web_md	NER model trained on Ontonotes 5.0
19. core_web_md+c	NER model trained on Ontonotes 5.0 + postprocessing
20. BTC	NER model trained on the Broad Twitter Corpus
21. BTC+c	NER model trained on the Broad Twitter Corpus + postprocessing
22. SEC	NER model trained on SEC-filings
23. SEC+c	NER model trained on SEC-filings + postprocessing
24. edited_core_web_md	NER model trained on Ontonotes 5.0 + alternative postprocessing
25. edited_core_web_md+c	NER model trained on Ontonotes 5.0 + alternative postprocessing
26. wiki_cased	Gazetteer (case-sensitive) using Wikipedia entries
27. wiki_uncased	Gazetteer (case-insensitive) using Wikipedia entries
28. multitoken_wiki_cased	Same as above, but restricted to multitoken entities
29. multitoken_wiki_uncased	Same as above, but restricted to multitoken entities
30. wiki_small_cased	Gazetteer (case-sensitive) using Wikipedia entries with non-empty description
31. wiki_small_uncased	Gazetteer (case-insensitive) using Wikipedia entries with non-empty description
32. multitoken_wiki_small_cased	Same as above, but restricted to multitoken entities
33. multitoken_wiki_small_uncased	Same as above, but restricted to multitoken entities
34. geo_cased	Gazetteer (case-sensitive) using the Geonames database
35. geo_uncased	Gazetteer (case-insensitive) using the Geonames database
36. multitoken_geo_cased	Same as above, but restricted to multitoken entities
37. multitoken_geo_uncased	Same as above, but restricted to multitoken entities
38. crunchbase_cased	Gazetteer (case-sensitive) using the Crunchbase Open Data Map
39. crunchbase_uncased	Gazetteer (case-insensitive) using the Crunchbase Open Data Map
40. multitoken_crunchbase_cased	Same as above, but restricted to multitoken entities
41. multitoken_crunchbase_uncased	Same as above, but restricted to multitoken entities
42. product_cased	Gazetteer (case-sensitive) using products extracted from DBPedia
43. product_uncased	Gazetteer (case-insensitive) using products extracted from DBPedia
44. multitoken_product_cased	Same as above, but restricted to multitoken entities
45. multitoken_product_uncased	Same as above, but restricted to multitoken entities
46. doclevel_voter	Considers all appearances of the same entity string in the document
47. doc_history_cased	Considers already introduced entities in the document (case-sensitive)
48. doc_history_uncased	Considers already introduced entities in the document (case-insensitive)
49. doc_majority_cased	Considers all entities in the document (case-sensitive)
50. doc_majority_uncased	Considers all majority labels in the document (case-insensitive)

Table 11: List of rules for the NCBI dataset. The rules are the same as the tagging rules in (Safranchik et al., 2020). Python implementations: `https://github.com/BatsResearch/safranchik-aaai20-code/blob/master/NCBI-Disease/train_generative_models.py`

Rule name	Description
1. CoreDictionaryUncased	AutoNER dictionary (biomedical entities)
2. CoreDictionaryExact	AutoNER dictionary (biomedical entities, exact match)
3. CancerLike	Heuristic detection of entities that are relevant to cancer
4. CommonSuffixes	Heuristic detection of entities that are relevant to common diseases
5. Deficiency	Heuristic detection of entities that are relevant to deficiencies
6. Disorder	Heuristic detection of entities that are relevant to disorders
7. Lesion	Heuristic detection of entities that are relevant to lesions
8. Syndrome	Heuristic detection of entities that are relevant to syndroms
9. BodyTerms	UMLS dictionary entries for terms that are relevant to body parts
10. OtherPOS	Heuristic detection of parts of speech that are not relevant to any disease
11. StopWords	Heuristic detection of stop words that are not relevant to any disease
12. Punctuation	Heuristic detection of punctiations that are not relevant to any disease

Table 12: List of rules for the WikiGold dataset. The Python implementation of WikiGold rules is provided in the "skweak" repo: `https://github.com/NorskRegnesentral/skweak/blob/670fcdec680930ce3e497886d06d61e6a1f2c195/examples/ner/conll2003_ner.py`

Rule name	Description
1. BTC	NER model trained on the Broad Twitter Corpus
2. core_web_md	NER model trained on Ontonotes 5.0
3. crunchbase_cased	Gazetteer (case-sensitive) using the Crunchbase Open Data Map
4. crunchbase_uncased	Gazetteer (case-insensitive) using the Crunchbase Open Data Map
5. full_name_detector	Heuristic function to detect full person names
6. geo_cased	Gazetteer (case-sensitive) using the Geonames database
7. geo_uncased	Gazetteer (case-insensitive) using the Geonames database
8. misc_detector	Heuristic detection of entities of type NORP, LANGUAGE, FAC or EVENT
9. wiki_cased	Gazetteer (case-sensitive) using Wikipedia entries
10. wiki_uncased	Gazetteer (case-insensitive) using Wikipedia entries
11. multitoken_crunchbase_cased	Same as above, but restricted to multitoken entities
12. multitoken_crunchbase_uncased	Same as above, but restricted to multitoken entities
13. multitoken_geo_cased	Same as above, but restricted to multitoken entities
14. multitoken_geo_uncased	Same as above, but restricted to multitoken entities
15. multitoken_wiki_cased	Same as above, but restricted to multitoken entities
16. multitoken_wiki_uncased	Same as above, but restricted to multitoken entities

Table 13: List of rules for the LaptopReview dataset. The rules are the same as the tagging rules in (Safranchik et al., 2020). Python implementations: `https://github.com/BatsResearch/safranchik-aaai20-code/blob/master/LaptopReview/train_generative_models.py`

Rule name	**Description**
1. CoreDictionary	AutoNER dict with entries of terms that are relevant to electronics
2. OtherTerms	Heuristic detection of laptop entities based on a pre-defined keyword list
3. ReplaceThe	Heuristic detection of laptop entities based on the “replace the” phrase
4. iStuff	Heuristic detection of laptop entities based on uppercase letters
5. Feelings	Heuristic detection of laptop entities based on common expressions
6. ProblemWithThe	Heuristic detection of laptop entities based on common expressions
7. External	Heuristic detection of laptop entities based on common hardware expression
8. StopWords	Heuristic detection of stop words that are not relevant to electronics
9. Punctuation	Heuristic detection of punctuation that are not relevant to electronics
10. Pronouns	Heuristic detection of pronouns that are not relevant to electronics
11. NotFeatures	Heuristic detection of terms that are not relevant to laptop features
12. Adv	Heuristic detection of adverbs that are not relevant to electronics

```python
def keyword_price(x):
    keywords_pos=["cheap", "reasonable", "inexpensive", "economical"]
    keywords_neg=["overpriced", "expensive", "costly", "high-priced"]
    if any(word in x.text.lower() for word in keywords_neg):
        return NEG
    if any(word in x.text.lower() for word in keywords_pos):
        return POS
    return ABSTAIN
```

Figure 4: Example of weak rule from the Yelp dataset (rule 6: keyword_price from Table 8).

```python
@labeling_function(pre=[textblob_sentiment])
def textblob_lf(x):
    if x.polarity < -0.5:
        return NEG
    if x.polarity > 0.5:
        return POS
    return ABSTAIN
```

Figure 5: Example of weak rule from the Yelp dataset (rule 1: textblob_lf from Table 8).

```python
def money_generator(doc):
    """Searches for occurrences of money patterns in text"""

    i = 0
    while i < len(doc):
        tok = doc[i]
        if tok.text[0].isdigit():
            j = i + 1
            while (j < len(doc) and (doc[j].text[0].isdigit() or doc[j].norm_ in data_utils.MAGNITUDES)):
                j += 1

            found_symbol = False
            if i > 0 and doc[i - 1].text in (data_utils.CURRENCY_CODES | data_utils.CURRENCY_SYMBOLS):
                i = i - 1
                found_symbol = True
            if (j < len(doc) and doc[j].text in
                    (data_utils.CURRENCY_CODES | data_utils.CURRENCY_SYMBOLS | {"euros", "cents", "rubles"})):
                j += 1
                found_symbol = True

            if found_symbol:
                yield i, j, "MONEY"
            i = j
        else:
            i += 1
```

Figure 6: Example of weak rule from the CoNLL dataset (rule 3: money_detector from Table 10). This rule heuristically detects entities that are relevant to money.

```python
def number_generator(doc):
    """Searches for occurrences of number patterns (cardinal, ordinal, quantity or percent) in text"""

    i = 0
    while i < len(doc):
        tok = doc[i]

        if tok.lower_ in data_utils.ORDINALS:
            yield i, i + 1, "ORDINAL"

        elif re.search("\\d", tok.text):
            j = i + 1
            while (j < len(doc) and (doc[j].norm_ in data_utils.MAGNITUDES)):
                j += 1
            if j < len(doc) and doc[j].lower_.rstrip(".") in data_utils.UNITS:
                j += 1
                yield i, j, "QUANTITY"
            elif j < len(doc) and doc[j].lower_ in ["%", "percent", "pc.", "pc", "pct", "pct.", "percents",
                                                    "percentage"]:
                j += 1
                yield i, j, "PERCENT"
            else:
                yield i, j, "CARDINAL"
            i = j - 1
        i += 1
```

Figure 7: Example of weak rule from the CoNLL dataset (rule 16: number_detector from Table 10). This rule heuristically detects entities that are relevant to numbers.

```python
class CancerLike(TaggingRule):
    def apply_instance(self, instance):
        tokens = [token.text.lower() for token in instance['tokens']]
        labels = ['ABS'] * len(tokens)

        suffixes = ("edema", "toma", "coma", "noma")

        for i, token in enumerate(tokens):
            for suffix in suffixes:
                if token.endswith(suffix) or token.endswith(suffix + "s"):
                    labels[i] = 'I'
        return labels
```

Figure 8: Example of weak rule from the NCBI dataset (rule 3: CancerLike from Table 11). This rule heuristically detects entities that are relevant to cancer.

```python
class StopWords(TaggingRule):

    def apply_instance(self, instance):
        labels = ['ABS'] * len(instance['tokens'])

        for i in range(len(instance['tokens'])):
            if instance['tokens'][i].lemma_ in stop_words:
                labels[i] = 'O'
        return labels
```

Figure 9: Example of weak rule from the NCBI dataset (rule 11: StopWords from Table 11). This rule heuristically detects stop words and assigns the 'O' tag to the corresponding tokens by assuming that they are not relevant to any disease.

```python
class Feelings(TaggingRule):
    feeling_words = {"like", "liked", "love", "dislike", "hate"}

    def apply_instance(self, instance):
        tokens = [token.text for token in instance['tokens']]
        labels = ['ABS'] * len(tokens)

        for i in range(len(tokens) - 2):
            if tokens[i].lower() in self.feeling_words and tokens[i +
                                                                   1].lower() == 'the':
                if instance['tokens'][i + 2].pos_ == "NOUN":
                    labels[i] = 'O'
                    labels[i + 1] = 'O'
                    labels[i + 2] = 'I'

        return labels
```

Figure 10: Example of weak rule from the LaptopReview dataset (rule 5: Feelings from Table 13). This rule heuristically detects entities that are relevant to laptop features based on keywords that express the user's feelings.

C Additional Results

Table 14 shows standard deviation results for all datasets, methods, and base models. The rightmost column responds the average standard deviation (AVG std) across tasks, which we also reported in Table 3.

Analysis of individual weak rules. Tables 15-21 show performance results for each weak rule for the datasets in WALNUT. We evaluate two different strategies for majority voting in case of an instance that is not covered by any rules: (1) "Strict" counts the instance as misclassified and (2) "Loose" assigns a random label to the instance. Most rules have very low F1 score while there are a few rules with a relatively high F1 score.

Figure 3 shows the precision-recall scatter plots for each weak rule individually. (We skip the scatter plot for GossipCop as it has just 3 rules.) Several rules have relatively high precision but most rules have very low recall.

Table 14: Standard deviation results on WALNUT.

Method	AGNews	IMDB	Yelp	GossipCop	CoNLL	NCBI	WikiGold	LaptopReview	AVG
					BiLSTM				
Full Clean	0.2	0.5	0.3	1.0	7.2	1.6	0.5	2.5	1.7
C	1.1	2.9	4.4	2.0	1.0	1.5	0.9	5.0	2.4
W	6.1	0.5	2.4	0.9	3.2	1.5	0.6	1.2	2.1
C+W	0.3	0.9	1.2	1.1	6.7	1.1	0.6	2.9	1.9
Snorkel	3.9	0.8	0.6	0.4	2.3	2.1	0.7	1.4	1.5
C+Snorkel	0.4	1.0	1.2	1.5	2.8	1.7	0.6	2.1	1.4
GLC	1.3	0.3	0.9	1.7	7.2	1.2	0.5	0.9	1.7
MetaWN	1.8	0.3	2.0	1.3	0.0	1.3	0.4	0.5	0.9
MLC	2.1	0.3	3.8	1.6	0.0	1.1	0.3	0.7	1.2
					DistilBERT				
Full Clean	0.2	0.3	0.3	1.5	0.4	0.4	0.7	3.5	0.9
C	6.2	6.8	7.4	6.4	3.6	2.0	1.3	1.7	4.4
W	3.9	1.1	1.5	1.1	0.9	1.2	0.3	2.0	1.5
C+W	0.6	0.2	0.9	1.2	0.8	1.4	0.4	3.6	1.1
Snorkel	3.0	1.6	0.6	0.5	1.1	1.5	0.4	2.2	1.4
C+Snorkel	0.7	0.5	2.0	0.8	0.9	1.9	0.4	3.6	1.4
GLC	2.8	0.5	1.5	2.0	2.1	1.8	0.2	1.5	1.5
MetaWN	1.6	1.3	0.8	2.2	1.8	1.4	0.3	0.6	1.3
MLC	2.5	0.6	0.7	1.6	1.2	1.8	0.4	2.3	1.4
					BERT				
Full Clean	0.1	0.5	0.2	1.0	0.6	0.5	1.0	1.8	0.7
C	0.9	8.1	5.2	1.8	1.3	0.8	1.3	2.5	2.7
W	2.7	0.5	1.1	2.2	1.2	2.8	0.9	1.8	1.7
C+W	0.4	0.6	1.4	1.5	0.9	1.4	0.8	1.5	1.1
Snorkel	2.3	3.7	1.3	0.9	1.3	3.5	1.0	1.3	1.9
C+Snorkel	1.0	0.5	0.6	0.6	1.6	1.7	0.8	2.6	1.2
GLC	1.6	0.8	2.2	2.8	2.4	1.2	0.4	1.2	1.6
MetaWN	1.1	1.0	1.0	2.4	1.6	0.5	0.3	1.4	1.2
MLC	2.0	0.8	1.3	1.4	2.1	2.8	0.2	0.4	1.4
					RoBERTa				
Full Clean	0.1	0.4	0.2	1.0	0.3	0.7	1.0	2.0	0.7
C	2.0	5.4	5.9	5.2	2.3	2.1	1.7	4.1	3.6
W	1.2	0.7	1.2	2.4	1.4	1.5	0.9	2.7	1.5
C+W	0.9	1.7	1.4	1.0	1.6	1.5	0.6	5.3	1.8
Snorkel	3.2	2.3	2.9	0.6	2.0	1.1	0.9	2.9	2.0
C+Snorkel	0.7	2.2	1.6	1.8	1.8	3.1	0.8	5.7	2.2
GLC	1.3	0.7	1.8	2.3	3.2	0.4	0.4	0.8	1.4
MetaWN	2.7	0.9	1.4	2.1	0.7	0.9	0.3	1.1	1.3
MLC	1.6	1.2	1.0	1.3	1.2	2.7	0.2	3.1	1.6
					BERT-large				
Full Clean	0.1	0.4	0.3	0.6	1.0	1.4	1.2	3.1	1.0
C	22.6	3.7	5.8	3.2	4.0	2.4	2.3	3.4	5.9
W	1.1	2.4	1.2	1.4	1.2	2.0	1.1	1.0	1.4
C+W	2.1	1.6	0.9	1.8	1.6	1.9	0.9	3.8	1.8
Snorkel	2.2	1.5	0.5	1.5	0.7	4.0	1.1	1.6	1.6
C+Snorkel	0.9	1.4	0.8	1.4	1.0	4.5	1.1	2.8	1.7
GLC	2.0	0.9	1.1	1.2	1.7	1.2	0.9	2.0	1.4
MetaWN	1.9	1.0	3.9	2.0	1.5	1.0	0.9	23.0	4.4
					RoBERTa-large				
Full Clean	0.07	0.33	0.16	0.59	0.7	0.7	0.8	2.0	0.7
C	1.8	9.6	7.8	1.0	1.5	1.2	0.7	4.7	3.5
W	0.8	0.7	0.5	2.7	2.0	4.1	1.8	2.9	1.9
C+W	1.2	1.6	1.6	2.6	1.9	1.3	0.7	5.0	2.0
Snorkel	0.8	2.5	2.5	1.9	1.2	2.9	1.9	4.4	2.3
C+Snorkel	2.1	2.5	1.5	2.3	2.5	3.1	0.7	2.8	2.2
GLC	1.7	1.0	2.1	1.2	28.4	1.2	0.8	3.8	5.0
MetaWN	2.0	16.6	3.0	15.6	1.4	1.5	0.6	2.9	5.5

Table 15: Performance of each rule on AGNews.

Rule	AG News unlabeled Prec	Rec	F1	train Prec	Rec	F1	validation Prec	Rec	F1	test Prec	Rec	F1
rule 1	0.179	0.078	0.109	0.179	0.114	0.140	0.182	0.077	0.108	0.180	0.079	0.110
rule 2	0.157	0.082	0.108	0.156	0.121	0.136	0.159	0.082	0.108	0.154	0.081	0.106
rule 3	0.162	0.093	0.118	0.162	0.134	0.147	0.160	0.094	0.118	0.166	0.094	0.120
rule 4	0.192	0.011	0.021	0.192	0.018	0.033	0.192	0.012	0.022	0.193	0.011	0.021
rule 5	0.187	0.064	0.095	0.189	0.090	0.122	0.190	0.067	0.099	0.188	0.068	0.099
rule 6	0.140	0.053	0.077	0.141	0.100	0.117	0.137	0.054	0.077	0.141	0.051	0.075
rule 7	0.161	0.052	0.079	0.163	0.096	0.121	0.163	0.050	0.077	0.163	0.051	0.078
rule 8	0.136	0.114	0.124	0.138	0.168	0.152	0.134	0.113	0.123	0.137	0.118	0.127
rule 9	0.152	0.007	0.014	0.153	0.011	0.020	0.149	0.007	0.013	0.154	0.008	0.014
Majority (strict)	0.649	0.426	0.512	0.814	0.812	0.812	0.645	0.424	0.509	0.650	0.429	**0.514**
Majority (loose)	0.618	0.620	0.617	0.814	0.812	0.812	0.611	0.613	0.610	0.618	0.620	0.618

Table 16: Performance of each rule on IMDB.

Rule	IMDB unlabeled Prec	Rec	F1	train Prec	Rec	F1	validation Prec	Rec	F1	test Prec	Rec	F1
rule 1	0.182	0.001	0.001	0.000	0.000	0.000	0.333	0.000	0.001	0.000	0.000	0.000
rule 2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
rule 3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
rule 4	0.497	0.405	0.446	0.502	0.478	0.489	0.505	0.404	0.448	0.513	0.423	**0.463**
rule 5	0.538	0.044	0.073	0.549	0.045	0.075	0.408	0.039	0.067	0.481	0.046	0.077
rule 6	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
rule 7	0.457	0.109	0.176	0.463	0.120	0.190	0.448	0.095	0.156	0.459	0.115	0.183
rule 8	0.655	0.006	0.012	0.644	0.004	0.009	0.630	0.008	0.015	0.667	0.008	0.015
Majority (strict)	0.495	0.426	0.457	0.749	0.745	0.745	0.501	0.423	0.458	0.511	0.448	**0.476**
Majority (loose)	0.708	0.707	0.706	0.749	0.745	0.745	0.710	0.708	0.708	0.740	0.739	0.739

Table 17: Performance of each rule on Yelp.

Rule	Yelp unlabeled Prec	Rec	F1	train Prec	Rec	F1	validation Prec	Rec	F1	test Prec	Rec	F1
rule 1	0.642	0.047	0.085	0.638	0.053	0.094	0.643	0.052	0.093	0.614	0.042	0.076
rule 2	0.214	0.029	0.051	0.221	0.036	0.063	0.213	0.028	0.050	0.239	0.031	0.054
rule 3	0.501	0.328	0.371	0.504	0.393	0.419	0.514	0.338	0.381	0.492	0.324	**0.367**
rule 4	0.498	0.064	0.114	0.485	0.081	0.139	0.501	0.069	0.121	0.491	0.066	0.117
rule 5	0.502	0.101	0.163	0.503	0.122	0.191	0.489	0.090	0.147	0.519	0.105	0.168
rule 6	0.426	0.035	0.065	0.433	0.046	0.083	0.417	0.036	0.066	0.398	0.036	0.066
rule 7	0.486	0.044	0.081	0.484	0.053	0.095	0.509	0.044	0.081	0.479	0.039	0.071
rule 8	0.553	0.049	0.085	0.556	0.060	0.103	0.515	0.049	0.086	0.553	0.053	0.092
Majority (strict)	0.508	0.389	0.411	0.762	0.700	0.692	0.515	0.392	0.415	0.498	0.381	**0.404**
Majority (loose)	0.710	0.677	0.663	0.762	0.700	0.692	0.719	0.683	0.671	0.706	0.672	0.659

Table 18: Performance of each rule on GossipCop.

	GossipCop								
	train			validation			test		
Rule	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**
rule 1	0.632	0.629	0.627	0.614	0.610	0.607	0.629	0.627	0.625
rule 2	0.648	0.622	0.604	0.643	0.620	0.604	0.658	0.630	0.613
rule 3	0.740	0.731	0.728	0.754	0.746	0.744	0.732	0.726	0.724
majority	0.758	0.732	0.725	0.757	0.728	0.721	0.760	0.740	**0.735**

Table 19: Performance of each rule on NCBI.

	NCBI								
	train			validation			test		
Rule	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**
rule 1	0.490	0.025	0.047	0.460	0.066	0.116	0.537	0.031	0.058
rule 2	0.514	0.017	0.034	0.140	0.010	0.019	0.349	0.016	0.030
rule 3	0.317	0.035	0.064	0.241	0.018	0.033	0.295	0.024	0.045
rule 4	0.875	0.219	0.350	0.911	0.118	0.208	0.807	0.172	**0.283**
rule 5	0.823	0.412	0.549	0.707	0.445	0.546	0.793	0.412	**0.542**
rule 6	0.678	0.037	0.071	0.794	0.035	0.066	0.667	0.030	0.057
rule 7	0.227	0.002	0.004	0.333	0.001	0.003	0.000	0.000	0.000
rule 8	0.250	0.001	0.001	0.000	0.000	0.000	0.000	0.000	0.000
rule 9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
rule 10	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
rule 11	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
rule 12	0.325	0.016	0.031	0.036	0.001	0.002	0.375	0.013	0.024
Majority	0.749	0.637	0.688	0.659	0.566	0.609	0.716	0.590	**0.647**

Table 20: Performance of each rule on WikiGold.

Rule	train Prec	train Rec	train F1	validation Prec	validation Rec	validation F1	test Prec	test Rec	test F1
rule 0	0.590	0.406	0.481	0.539	0.399	0.459	0.591	0.397	0.475
rule 1	0.593	0.538	0.564	0.597	0.558	0.577	0.624	0.557	0.589
rule 2	0.252	0.059	0.095	0.235	0.049	0.081	0.229	0.059	0.093
rule 3	0.226	0.060	0.095	0.193	0.049	0.078	0.211	0.061	0.095
rule 4	0.621	0.091	0.158	0.596	0.086	0.150	0.633	0.101	0.175
rule 5	0.776	0.137	0.233	0.814	0.147	0.249	0.886	0.104	0.186
rule 6	0.773	0.137	0.233	0.814	0.147	0.249	0.886	0.104	0.186
rule 7	0.576	0.092	0.159	0.542	0.080	0.139	0.587	0.099	0.169
rule 8	0.558	0.030	0.058	0.471	0.025	0.047	0.684	0.035	0.066
rule 9	0.547	0.030	0.058	0.471	0.025	0.047	0.684	0.035	0.066
rule 10	0.875	0.020	0.038	0.857	0.037	0.071	1.000	0.024	0.047
rule 11	0.862	0.020	0.038	0.857	0.037	0.071	1.000	0.024	0.047
rule 12	0.885	0.177	0.295	0.864	0.215	0.344	0.857	0.176	0.292
rule 13	0.869	0.178	0.296	0.855	0.218	0.347	0.825	0.176	0.290
rule 14	0.780	0.352	0.485	0.803	0.387	0.522	0.781	0.315	0.449
rule 15	0.758	0.353	0.482	0.768	0.387	0.514	0.727	0.312	0.437
Majority	0.490	0.564	0.524	0.488	0.558	0.521	0.490	0.560	0.522

Table 21: Performance of each rule on LaptopReview.

Rule	train Prec	train Rec	train F1	validation Prec	validation Rec	validation F1	test Prec	test Rec	test F1
rule 1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
rule 2	0.679	0.595	0.634	0.656	0.584	0.618	0.722	0.512	**0.599**
rule 3	0.667	0.003	0.006	1.000	0.004	0.008	0.500	0.003	0.006
rule 4	0.500	0.006	0.012	0.400	0.009	0.017	0.750	0.009	0.018
rule 5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
rule 6	0.423	0.006	0.011	0.467	0.015	0.029	0.000	0.000	0.000
rule 7	1.000	0.001	0.002	0.500	0.002	0.004	0.000	0.000	0.000
rule 8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
rule 9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
rule 10	0.333	0.001	0.001	1.000	0.002	0.004	0.000	0.000	0.000
rule 11	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
rule 12	0.735	0.013	0.026	0.800	0.009	0.017	0.250	0.006	0.012
Majority	0.671	0.609	0.638	0.644	0.599	0.621	0.706	0.521	**0.600**