Token-Level Metrics for Detecting Incorrect Gold Annotations in Named Entity Recognition

Elena Merdjanovska
Humboldt-Universität zu Berlin
Science of Intelligence
elena.merdjanovska@hu-berlin.de

Alan Akbik
Humboldt-Universität zu Berlin
Science of Intelligence
alan.akbik@hu-berlin.de

Abstract

Annotated datasets for supervised learning tasks often contain incorrect gold annotations, i.e. label noise. To address this issue, many noisy label learning approaches incorporate metrics to filter out unreliable samples, for example using heuristics such as high loss or low confidence. However, when these metrics are integrated into larger pipelines, it becomes difficult to compare their effectiveness, and understand their individual contribution to reducing label noise. This paper directly compares popular sample metrics for detecting incorrect annotations in named entity recognition (NER). NER is commonly approached as token classification, so the metrics are calculated for each training token and the incorrect ones are flagged by defining metrics thresholds. We compare the metrics based on (i) their accuracy in detecting the incorrect labels and (ii) the test scores when retraining a model using the cleaned dataset. We show that training dynamics metrics work the best overall. The best metrics effectively reduce the label noise across different noise types. The errors that the model has not yet memorized are more feasible to detect, and relabeling these tokens is a more effective strategy than excluding them from training.[1]

1 Introduction

State-of-the-art approaches for some NLP tasks still require supervision in the form of labeled training data (Zaratiana et al., 2023). One example is named entity recognition (NER), the task of identifying and classifying named entities in text. For NER we need sentences in which named entities are marked and their correct type is assigned. However, available datasets for NER are affected by *label noise*, meaning that a certain percentage of labels are incorrect (Wang et al., 2019; Reiss et al., 2020; Rücker and Akbik, 2023).

Such erroneous annotations can deteriorate model quality in machine learning tasks, including NER. One promising approach involves selecting a subset of clean samples for training (Merdjanovska et al., 2024) and excluding the ones with an error. This requires a robust way to detect the incorrectly labeled training samples.

Noisy label learning approaches often incorporate measures to select reliable samples as part of complex pipelines, for example heuristics such as low loss or high confidence (Chen et al., 2020; Zhu et al., 2023a). Some studies (Swayamdipta et al., 2020; Siddiqui et al., 2022), propose new measures to characterize dataset samples based on their learning dynamics. However, when these measures are integrated into larger pipelines, it becomes difficult to isolate their impact, compare their effectiveness, and understand their individual contributions to reducing label noise. Furthermore, many existing approaches rely on computationally expensive techniques such as consistency checks, self-training, and ensemble methods (Wang et al., 2019; Liang et al., 2020). While effective, their efficiency is often a limiting factor.

In this paper, we focus on evaluating sample metrics that can identify label errors from a single training run. We include both *static metrics*, calculated from a single epoch and *training dynamics metrics*, which aggregate information over multiple epochs. Additionally, we assess the potential of *layer-wise metrics*, which have not been utilized for annotation error detection before. Our goal is to assess the potential of these metrics as diagnostic tools for detecting mislabeled samples and to directly compare them.

NER is commonly approached as a token classification task, where each token is assigned a BIO tag, as illustrated in Figure 1. The confidence metric shown is the softmax probability of the observed label, averaged across epochs (Swayamdipta et al., 2020) and is calculated for each token in the train-

[1]GitHub repository: https://github.com/elenamer/TokenMetrics

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 15292–15304
November 4-9, 2025

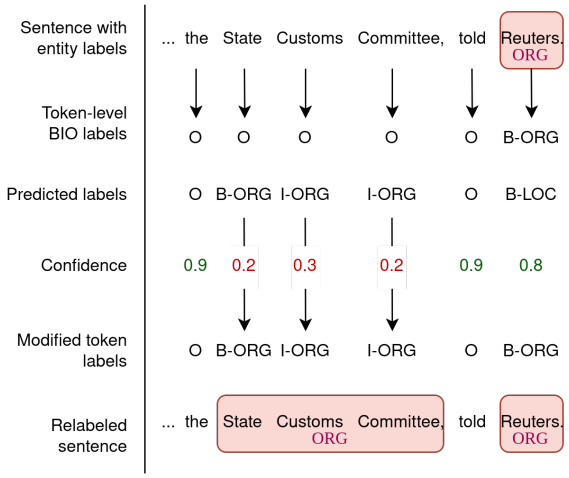

Figure 1: Example sentence demonstrating a missing mention error of type organization (ORG). The entities are represented as token-level BIO tags. The *confidence* metric is calculated for each token, and all tokens < 0.4 are flagged as errors. The flagged tokens are later relabeled using the model's predictions and the error in the original sentence is corrected.

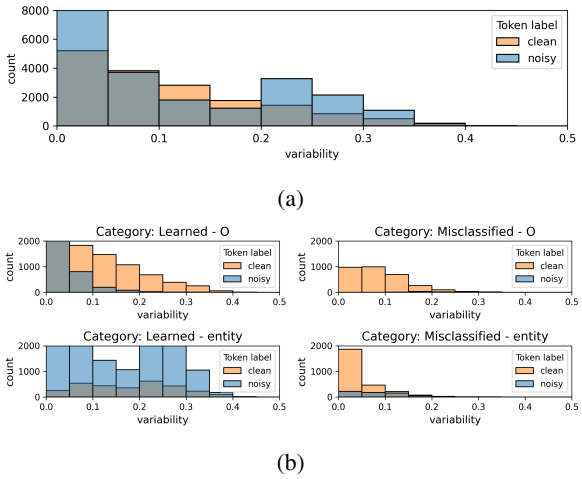

Figure 2: Histograms of the distribution of *variability*, where noisy and clean tokens are shown in different colors. Considering different types of tokens separately enables a better discrimination between clean and noisy tokens, based on *variability* values.

ing set. As illustrated, using confidence as an indicator for noise entails defining a threshold (for example < 0.4), and then flagging all samples below that threshold as erroneous. Additionally, the flagged tokens can be relabeled with the model's predicted tags. There exist other metrics besides confidence, and we want to know which one is best.

NER covers two subtasks: mention detection (tagging a token as either O or as belonging to a mention) and mention classification (tagging a token with the correct entity type). Due to this, it is challenging to define universal sample selection rules. Instead, we propose a disaggregated approach, where we handle four different categories of tokens independently. Figure 2 shows the separation between the correct and incorrect tokens, both by handling all tokens at once in Figure 2a and with our category approach in Figure 2b.

In this paper, we evaluate the use of sample metrics for detecting erroneous tokens in noisy NER datasets. Following are our main contributions:

- We propose a token category approach, which offers better separation than dealing with all tokens at once. This approach also provides better insights into which metrics are suitable for detecting different types of NER errors.
- We define novel metrics based on layer-wise transformer predictions.
- We show that training dynamics metrics are the best overall, however the simpler static ones are also suitable for some categories.
- We show that relabeling the tokens detected as potential errors by our method successfully reduces the noise share by 3.3 percentage points on average. Additionally, we show that automatically relabelling these tokens is more effective than filtering them out from the training process.

2 Token-Level Metrics

We explore three types of token-level metrics: static metrics, training dynamics metrics, and layer agreement metrics. These metrics characterize each training sample based on the model's outputs.

2.1 Static Metrics

Static metrics are computed at a single epoch, without considering earlier training steps. This contrasts with training dynamics metrics, which span the full training history.

Cross-entropy: Measures the divergence between predicted and true label distributions. It is used as training loss for classification, with lower values typically indicating correct predictions. Deviations from this may signal label noise.

Maximum softmax probability (MSP): The highest softmax score among class predictions, commonly used to estimate confidence (Zou and

Caragea, 2023; Yu et al., 2021). Higher MSP suggests higher confidence; low values suggest uncertainty among classes.

Prediction margin: The difference between the top two softmax scores. A small margin indicates similar confidence in multiple labels and suggests uncertainty (Pleiss et al., 2020; Joshi et al., 2009).

Predictive entropy: The entropy of softmax outputs reflects uncertainty (Song et al., 2019). High entropy implies dispersed predictions across classes, signaling low confidence.

2.2 Training Dynamics Metrics

These metrics use information from the full training trajectory. Some are based on counts (e.g. correctness), while others aggregate a static metric across epochs.

Confidence: The average softmax probability assigned to the true label over training epochs (Swayamdipta et al., 2020). Low confidence suggests the model resists learning the label, possibly due to label error.

Variability: The standard deviation of the true label's softmax probability across epochs (Swayamdipta et al., 2020). It captures how prediction certainty changes over time.

Correctness: The fraction of epochs in which the model predicts the correct label (Swayamdipta et al., 2020). Like confidence, it reflects the model's trust in the label.

Iteration learned: Counts how many consecutive epochs the model's prediction remains unchanged (Baldock et al., 2021; Toneva et al., 2019). Stability in prediction is interpreted as confidence.

MILD Metric: Extends iteration learned by combining the number of epochs to memorize and forget a sample (Hu et al., 2024). It captures both learning and forgetting dynamics.

Prediction history entropy (entropy-history): Entropy over predicted labels across epochs (He et al., 2024; Chen et al., 2021). It reflects how consistently the model predicts the same label.

2.3 Layer Agreement Metrics

sInstead of relying solely on the final layer's output, these metrics consider predictions from each transformer block. Following Baldock et al. (2021), who proposesprediction depth as an example difficulty measure, we also define other metrics that probe prediction consistency and model complexity. See Appendix D for the training setup of the layer-wise classifiers.

	Metric	Time	Space
static	cross-entropy	c	0
	MSP	c	0
	prediction-margin	c	0
	predictive-entropy	c	0
train. dynamics	confidence	c	1
	variability	c	1
	correctness	c	1
	iteration-learned	c	1
	MILD	$c \times e$	e
	entropy-history	$c \times e$	c
layer agreem.	prediction-depth	$c \times L$	0
	first-layer	$c \times L$	0
	agreement-predicted	$c \times L$	0
	agreement-true	$c \times L$	0
	layer-entropy	$c \times L$	0

Table 1: Computational complexity of the token metrics, where c is the number of classes, e is the number of previous epochs and L is the number of transformer layers. It should be noted that the compexity of calculating the predicted class, i.e. finding the maximum of the softmax values, is already c.

Prediction depth: The index of the earliest layer after which all subsequent layers give the same prediction. Lower values mean fewer layers are needed — suggesting a simpler or cleaner example. Higher values implies higher model complexity is needed, which could indicate noise.

First-layer with correct prediction (first-layer): The first layer that outputs the correct label. Inspired by the confidence metric, it tracks how early the model starts predicting correctly.

Prediction agreement count (agreement-predicted): The total number of layers that agree with the final prediction, regardless of order. Unlike prediction depth, layers don't need to be consecutive.

Correct agreement count (agreement-true): Counts how many layers predict the correct label, without requiring agreement with the final layer.

Layer entropy: Measures entropy over all layer predictions, capturing disagreement among layers.

2.4 Computational Complexity

The metrics' estimated computational complexity for one data example is presented in Table 1, where c is the number of classes, e the number of previous epochs and L the number of transformer layers. The time complexity refers to the number of operations, and space complexity denotes the number of variables that need to be saved between two training epochs.

It is important to note that the complexity of sim-

ply calculating the predicted class is already c. This means that all static and training-dynamics metrics have no added time complexity, adding only a single operation per epoch. The static metrics also do not add space complexity, while most training-dynamics metrics require one variable to be saved between epochs. This number is larger for MILD and entropy-history, which need an array of e and c elements respectively. In practice, the added space complexity is negligible as most labeled datasets easily fit into memory. Layer-agreement metrics change the complexity more significantly, because the prediction needs to be calculated for each layer, resulting in $c \times L$ time complexity and zero space complexity.

3 Approach

In this section, we present our approach for cleaning noisy NER data using token-level metrics. Section 3.1 describes how we select the most effective metrics and parameter settings. Section 3.2 introduces a disaggregated method that separates tokens into four categories to improve error detection. Finally, Section 3.3 outlines two strategies – relabeling and masking – for handling detected label errors. The full pipeline is illustrated in Figure 3.

3.1 Selecting the Best Metric

To quantify the effectiveness of various metrics in identifying mislabeled samples, we calculate the $F_{0.5}$ score on the binary classification task of discriminating clean and noisy samples. The ground truth is the true label quality, while the predicted label is determined by a threshold on the metric (e.g., flagging tokens with variability > 0.1 as noisy according to Figure 2b, Learned-*O* category).

We use $F_{0.5}$ which gives more weight to precision over recall—crucial for filtering, where mistakenly removing clean data must be minimized. Details on how we compute detection scores are provided in Appendix A.

We evaluat each metric across different thresholds and training epochs. We select the best-performing metric and configuration by maximizing the $F_{0.5}$ score, calculated on the training set. Note that this process requires access to clean labels for the training set, used to determine the ground truth label quality, so this procedure is not applicable to datasets with only noisy annotations. Our goal, therefore, is to derive generalizable findings on which metrics perform best, rather than repeating the metric selection for each dataset.

ID	Category	Correctness of predicted label	Observed label
1	Learned *O*	Correct	O
2	Misclassified *O*	Incorrect	O
3	Learned *entity*	Correct	B-PER, I-LOC...[2]
4	Misclassified *entity*	Incorrect	B-PER, I-LOC...[2]

Table 2: Overview of the token categories. The correctness of the *predicted label* is in reference to the *observed label*, which means the predicted label is correct if it matches the observed label. The *observed label* is the noisy training label.

3.2 Token Categories

NER is typically framed as a token classification task using BIO tags, involving two subtasks: mention detection (distinguishing "O" from entity tokens) and mention classification (assigning entity types to "non-O" tokens).[2]

Training samples are individual tokens, and our goal is to identify incorrectly labeled ones. This differs from standard classification due to the sparsity and class imbalance inherent in NER. We find it helpful to divide tokens into four categories based on (1) whether the model predicts the correct class, and (2) whether the true label is "O" or an entity type[3]. This improves detection performance by reducing variability within each group and addressing class imbalance. The categories are summarized in Table 2, and visualized in Figure 2. We select the best metric separately for each category.

3.3 Label Modifications

Once potentially noisy tokens are flagged, we apply either *masking* or *relabeling* on them.

In the masking approach, we assign flagged tokens a special 'MASK' label, which is excluded from training loss computations. This effectively removes them from training while preserving sequence structure — important in NER due to the sequential labeling format.

In the relabeling approach, we assign the flagged token a new label – the model's prediction. This is only applicable to the two Misclassified categories. For Misclassified-*O* tokens, relabelling changes O tokens into B- or I- tokens, which usually introduces a new entity mention. In this case, in addition to the flagged token, we also modify the other

[2]E.g., for entity types *person* and *organization*, BIO labels include *B-person*, *I-person*, *B-organization*, etc.

[3]The true label is the noisy one obeserved by the model.

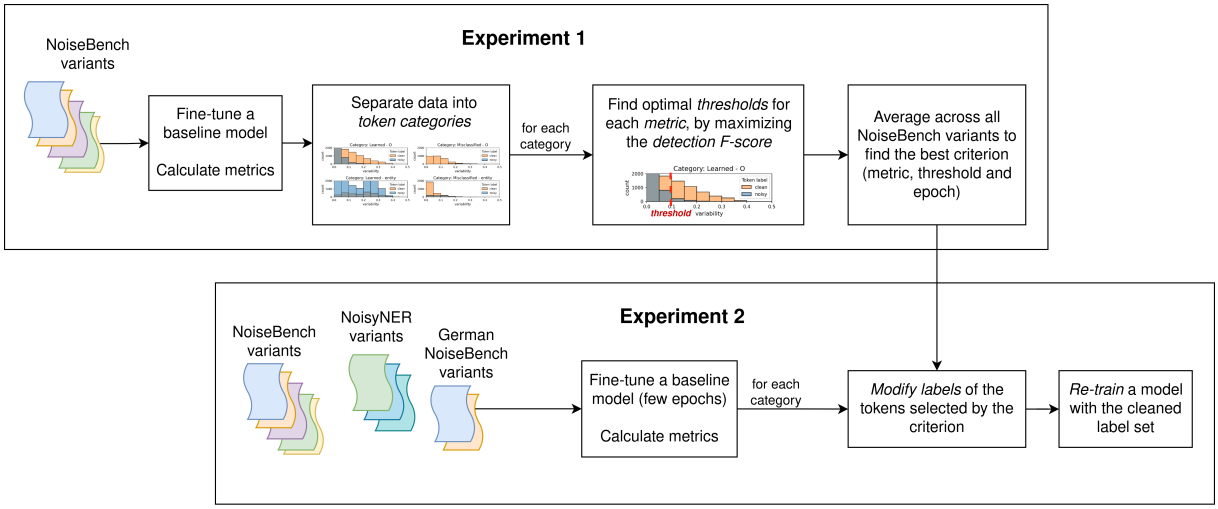

Figure 3: Overview of the experimental pipeline. This figure shows how the components of our approach fit together and the setup of Experiments 1 and 2.

	Example 1			Example 2		
orig.	B-LOC	**O**	O	B-PER	**I-PER**	O
pred.	B-LOC	**I-LOC**	I-LOC	B-ORG	**I-ORG**	I-ORG
relab.	B-LOC	**I-LOC**	I-LOC	B-PER	**B-ORG**	I-ORG

Table 3: Relabelling examples with original (noisy), predicted and relabelled token sequences. The flagged token in each sequence is **boldfaced**.

tokens from the same mention that follow it. See Example 1 in Table 3. If the relabelled sequence is invalid, we make it valid by converting I- to B-tags. See Example 2 in Table 3.

Because tokens in the Misclassified categories have not been memorized yet, the model's prediction likely reflects the correct label. This is likely due to the model's pretraining knowledge, making it possible to recover plausible labels early in training, before it overfits to noisy annotations. This makes the Misclassified noisy tokens more amenable to correction than tokens that have already been memorized with the noisy label – the Learned categories.

After applying label modifications, we retrain the baseline model on the cleaned dataset to assess downstream performance.

4 Experiment 1: Evaluating Selection Performance

In this experiment, we determine which metrics are most effective as noise indicators. We also compare the ideal parameter configuration for each noise type with a single configuration that performs well on average. This allows us to assess both the maximum selection potential and the generalizability of metric performance.

4.1 Dataset(s)

We use the NoiseBench[4] (Merdjanovska et al., 2024) dataset, consisting of 4,879 training sentences and 3,427 test sentences[5]. This dataset includes a validation set of 1006 sentences which we do no use. This dataset is based on the English CoNLL-03 and it covers multiple noisy-labeled variants of the training set, with six types of real label noise: expert, crowdsourcing, distant supervision, weak supervision and LLM labels. The noise types and their respective shares are listed in Table 5, under the *Original* column.

4.2 Experimental Setup

The experimental pipeline is illustrated in Figure 3. The final step involves selecting the optimal criteria (threshold and epoch) for each metric. The baseline model parameters and the threshold ranges for each metric are given in Appendix B.

We compute two types of $F_{0.5}$ scores. The first one – *average* score – is obtained by selecting a single parameter setting per metric and averaging the $F_{0.5}$ across all NoiseBench variants.

The second one – *optimal average* score – is calculated by selecting different optimal parameter settings per noise variant, then averaging the $F_{0.5}$ scores. While the *optimal* average shows the maximum possible performance of each metric,

[4]License: MIT

[5]Excluding document separators.

	Learned-*O*		Misclassified-*O*		Learned-*entity*		Misclassified-*entity*	
	avg.	optimal avg.	avg.	optimal avg.	avg.	optimal avg.	avg.	optimal avg.
cross-entropy	0.535	0.630	0.892	0.931	0.345	0.381	0.678	0.721
MSP	0.536	0.629	0.892	0.924	0.343	0.379	0.671	0.712
prediction-margin	0.534	0.629	0.892	0.927	0.349	0.381	0.670	0.711
predictive-entropy	0.532	0.633	0.869	0.902	0.347	0.415	0.673	0.715
confidence	0.591	**0.730**	**0.908**	0.940	0.464	0.490	**0.681**	**0.727**
variability	**0.628**	0.722	0.904	0.918	0.357	0.409	0.676	0.727
correctness	0.619	0.679	0.883	0.895	**0.477**	**0.494**	0.679	0.722
iteration-learned	0.619	0.652	0.892	0.920	0.471	0.494	0.657	0.689
MILD	0.619	0.679	0.892	**0.942**	**0.477**	**0.494**	0.679	0.722
entropy-history	0.608	0.640	0.806	0.823	0.436	0.465	0.669	0.700
prediction-depth	0.547	0.635	0.803	0.864	0.319	0.363	0.599	0.640
first-layer	0.442	0.482	0.799	0.898	0.261	0.271	0.590	0.629
agreement-predicted	0.559	0.640	0.821	0.903	0.310	0.332	0.604	0.643
agreement-true	0.559	0.640	0.821	0.913	0.310	0.332	0.621	0.660
layer-entropy	0.541	0.620	0.808	0.854	0.301	0.322	0.596	0.621

Table 4: Comparison of *optimal* and *average* detection F-scores. **Boldfaced** are the highest scores in each column, all of which come from training dynamics-based metrics. Largest gap between *optimal* and *average* is in Learned-*O*.

the *average* score reflects how well our general approach can approximate that potential.

4.3 Results

Table 4 shows the highest $F_{0.5}$ scores for each metric and category, reporting the *average* and *optimal* scores. The parameter settings used to obtain the *average* scores are provided in Appendix C.

Large gap to optimal for *Learned-O*. Several metrics, including correctness, variability, iteration-learned, and MILD, achieve *average* $F_{0.5}$ scores near 0.62, indicating that missing mention errors memorized by the model are moderately detectable using different strategies. The gap to the *optimal* scores is however large, where confidence achieves the highest *optimal* average score at 0.73, followed by variability at 0.722. Variability comes closer to this potential with a higher *average* score, making it more practical, however this gap is large at 10 percentage points.

***Misclassified-O*: High scores, low gap to optimal.** Most metrics from the static and training dynamics groups perform well on this category, with *average* scores around 0.9 and *optimal* scores near 0.94—a much smaller gap than for Learned-*O*. This supports the idea that such errors are easier to detect before memorization. Appendix C shows that the thresholds for many metrics (e.g., MSP > 0) result in flagging nearly all samples. This means that simply detecting all tokens yields an $F_{0.5}$ of 0.89, close to the confidence-based maximum of 0.91.

Low scores, poor detection for *Learned non-O*. Table 9 shows low *average* $F_{0.5}$ scores for this category, with a maximum of 0.48. At this level, filtering could introduce more errors than it removes. This confirms that once false positive mention errors are memorized, token-level features fail to identify them. Figure 2b illustrates this poor separation. The *optimal* scores, barely reaching 0.5, suggest little untapped potential.

Low gap to optimal for *Misclassified non-O*. Static and training dynamics metrics perform similarly here, with *average* scores up to 0.68 and *optimal* scores around 0.727—indicating a low gap. Appendix C shows that naive approaches, like relabeling all tokens where prediction-margin > 0, yield $F_{0.5}$ scores of 0.67—nearly matching the best *average* value.

Discussion on metric types and categories. Training dynamics metrics perform the best overall, while simpler static metrics work well for Misclassified-*O* and Misclassified-*entity* categories, where errors have not been memorized. In contrast, layer-wise metrics perform poorly. The gap between *average* and *optimal* scores is the largest for Learned-*O* and smaller for the other categories. Overall, token-level metrics effectively detect errors in Misclassified-*O*, Misclassified-*entity*, and Learned-*O*, but not in Learned-*entity*.

5 Experiment 2: Evaluating Re-training Performance

This experiment evaluates how effective the best metrics from Experiment 1 are in cleaning noisy annotations and how much test scores improve when re-training with these modified annotations, com-

		% Noise shares ↓			% *correct* ↑			
		Original	Misclassified-*O* relabel	Misclassified-*entity* relabel	Misclassified-*O* mask	Misclassified-*O* relabel	Misclassified-*entity* mask	Misclassified-*entity* relabel
NoiseBench	Expert	**5.5**	5.7±0.0	6.3±0.1	72.2	78.9	44.2	74.3
	Crowd++	15.3	**10.8±0.3**	16.0±0.2	99.0	87.2	47.4	72.0
	Distant	31.3	**26.5±0.6**	32.3±0.2	99.8	93.1	51.3	52.2
	Crowd	36.6	**31.7±0.3**	37.0±1.2	99.8	83.0	51.4	74.1
	Weak	40.4	38.0±0.3	**31.1**±0.7	93.0	63.2	79.5	74.0
	LLM	45.6	41.7±0.2	**34.6±0.3**	93.4	81.6	89.8	82.1
	German Expert	**16.2**	17.8±0.2	16.9±0.1	14.1	86.8	58.3	62.8
	German LLM	54.0	51.3±0.3	**49.1±0.2**	56.6	88.1	87.7	78.2
NoisyNER	NoisyNER 1	72.0	**70.3±1.3**	72.6±0.2	97.2	89.2	21.2	53.8
	NoisyNER 2	61.0	**59.3±0.9**	61.2±0.3	91.8	94.3	20.4	45.0
	NoisyNER 3	66.0	**63.4±0.7**	65.9±0.4	91.9	80.1	56.7	64.2
	NoisyNER 4	60.0	**58.0±0.6**	59.7±0.1	95.2	95.7	20.9	48.1
	NoisyNER 5	56.0	**52.3±0.6**	54.5±0.8	93.5	84.7	56.2	62.2
	NoisyNER 6	54.0	**50.0±0.9**	52.2±0.4	93.1	80.0	47.0	50.4
	NoisyNER 7	46.0	**42.0±1.3**	44.8±0.2	92.9	84.8	51.9	51.0

Table 5: Noise share and %correct after the modifications. The first column shows the original noise share in terms of $100 - \%F1$. The second and third columns show the noise shares after relabeling the flagged tokens in Misclassified-*O* and Misclassified-*entity* respectively. **Bolded** are the best (lowest) noise shares in each row. The last four columns show the accuracy of masking and relabeling. Accuracies over 80% are underlined.

pared to using the original noisy sets.

5.1 Datasets

We extend the evaluation to two additional datasets: the German variant of NoiseBench (Merdjanovska et al., 2024) and the Estonian NoisyNER[6] (Hedderich et al., 2021), alongside the English NoiseBench used in Experiment 1. The German NoiseBench is based on the German CoNLL-03 dataset and consists of 10,367 training sentences and 3,005 test sentences[7]. It covers two noise types: expert and LLM labels. NoisyNER covers multiple levels of distant supervision noise by varying the amounts of heuristics during the automatic annotation process and provides seven sets of noisy labels. As the original dataset does not define training and test splits, we split it 80/10/10, resulting in 11,374 training and 1,433 test sentences. NoiseBench and NoisyNER also have noisy validation splits[8], which we do not use in our experiments. Table 5 (Original column) shows an overview of noise types and their shares for each dataset.

Since the German NoiseBench and NoisyNER datasets were not part of the metric selection process, they allow us to assess how well the selected metrics and parameter sets generalize.

[6]License: CC-BY-NC for data and CC-BY-4.0 for labels.
[7]Excluding document separators.
[8]The German NoiseBench and NoisyNER validation sets consist of 1,785 and 1,479 noisy sentences respectively.

5.2 Experimental Setup

Figure 3 shows the pipeline. For each category, we use the best metric and parameter combination determined on the English NoiseBench training data to flag potentially noisy tokens. We then modify their labels via masking or relabeling and retrain the model on the modified data.

We first evaluate label modifications by comparing noise shares before and after changes. Noise is measured as $100 - \%F1$, with F1 calculated between noisy and clean labels (Merdjanovska et al., 2024). Since this measure applies only to relabeling, we also report the proportion of correctly modified tokens for both relabeling and masking.

For the main evaluation, we measure test F1 scores after re-training on the modified datasets and compare them to the same baseline on the original datasets (see Appendix B). The test sets consist of clean labels, which is the standard setting for learning with noisy labels. We do this separately for each category, and for masking and relabeling where applicable. We also compare our best test scores with those from prior work for each dataset.

In addition to evaluating categories individually, we also test a combined modification strategy (referred to as *Combined* in the results table) that merges three categories, excluding Learned-*entity* due to its weak selection performance.

		Baseline	Learned-*O* mask	Misclassified-*O* mask	Misclassified-*O* relabel	Learned-*entity* mask	Misclassified-*entity* mask	Misclassified-*entity* relabel	Combined
NoiseBench	Expert	90.39±0.37	**90.5±0.1**	90.4±0.2	90.5±0.2	89.7±0.2	89.9±0.0	90.3±0.2	89.8±0.1
	Crowd++	87.13±0.78	88.3±0.3	88.0±0.3	**88.4±0.2**	85.7±1.0	85.8±0.3	86.4±0.2	87.4±0.2
	Distant	69.66±0.45	73.0±0.8	72.7±0.7	**74.4±0.4**	67.2±0.2	69.1±0.3	69.2±0.7	73.3±0.4
	Crowd	71.19±0.63	73.4±0.2	73.1±0.1	**74.3±0.4**	66.0±1.0	67.6±1.8	71.5±0.7	73.4±0.5
	Weak	64.94±0.09	64.8±0.3	64.6±0.1	65.5±0.3	66.6±0.6	**67.4±0.4**	65.5±0.3	65.1±0.6
	LLM	61.79±0.29	61.9±0.0	62.3±0.6	61.9±0.2	64.2±0.7	**64.6±0.4**	61.5±0.2	61.8±0.6
	German Expert	79.73±0.41	79.1±0.4	79.5±0.4	79.4±0.3	79.8±0.2	**79.9±0.3**	79.8±0.1	78.7±0.5
	German LLM	55.41±0.46	57.4±0.3	57.7±0.6	57.6±0.2	57.4±0.5	**57.9±0.5**	55.3±0.1	57.6±0.5
NoisyNER	NoisyNER 1	32.76±0.13	**36.1±0.3**	33.8±0.4	34.3±0.9	30.8±0.6	31.5±1.0	32.6±0.5	35.8±0.9
	NoisyNER 2	41.11±0.36	**44.1±0.6**	42.2±0.9	42.5±1.5	40.5±0.4	40.4±0.5	42.3±0.5	44.0±1.2
	NoisyNER 3	39.95±0.56	42.4±0.5	41.4±0.5	**43.1±1.3**	40.0±0.7	40.3±1.0	40.5±0.5	42.8±0.4
	NoisyNER 4	42.33±0.86	**45.4±0.7**	42.8±0.3	43.1±0.9	41.1±1.1	41.7±0.2	42.3±0.7	45.2±0.9
	NoisyNER 5	50.21±0.80	52.8±0.8	50.4±0.8	**53.1±0.6**	49.7±0.3	50.3±1.3	50.4±0.9	51.7±0.3
	NoisyNER 6	51.73±0.22	54.1±0.3	53.0±0.9	**55.3±1.3**	52.3±1.1	51.7±0.8	51.8±0.6	53.7±0.4
	NoisyNER 7	56.57±0.18	59.5±0.7	59.8±1.2	**62.5±1.9**	58.2±0.5	57.0±0.9	56.4±0.1	60.3±0.2

Table 6: F1-scores on the test set, after retraining with a modified dataset using the best-performing metrics. The columns show the categories and modifications and the rows show the dataset variants. **Bolded** are the highest scores in each row.

5.3 Results - Noise Shares

Noise shares decrease after relabeling. Table 5 (left) shows that relabeling successfully reduces noise share across all variants except the Expert ones. These variants already have low noise levels, making it harder to detect label errors.

Results align with noise characteristics. Relabeling Misclassified-*O* is generally more effective than relabeling Misclassified-*entity*, for all dataset variants but Weak, LLM, and German LLM (not considering the Expert variants). These three noise variants have mostly false positive errors (Merdjanovska et al., 2024), which fall into the Misclassified-*entity* category — so fixing this category has more impact. In contrast, Crowd++, Crowd, and Distant have mostly missing mention errors, falling into the Misclassified-*O* – so more errors are corrected when this category is relabeled.

Challenges in comparing masking and relabeling. The right half of Table 5 shows the percentage of correctly modified tokens with masking and relabeling. For masking, this is the proportion of masked tokens that were actually noisy. For relabeling, it is the share of noisy tokens whose new labels are correct. Directly comparing the two methods is difficult. Masking removes data points, while relabeling may introduce new errors if the new labels are wrong. Ideally, we want to avoid data loss, but masking requires it. When relabeling accuracy is low, it can do more harm than good. While this analysis is useful, final re-training performance is the best measure of effectiveness.

High accuracies for Misclassified-*O*. From the underlined values (>80%), we see that masking has very high accuracy, meaning a lot of the flagged tokens are indeed noisy. This is especially true for Misclassified-*O*, and much less for Misclassified-*entity* (only in LLM and German LLM).

5.4 Results - Retraining

Overall. As shown in Table 6, targeted modifications improve final test scores. For most datasets and variants, the improvement exceeds the baseline standard deviation, indicating statistical significance. The only exceptions are the Expert variants.

Comparing noise types, features, and categories. Distant and NoisyNER datasets, both involving distant supervision, show the largest improvements. Among token categories, Misclassified-*O* yields the most consistent performance gains, confirming its utility for identifying erroneous tokens. Learned-*O* is most effective on the NoisyNER datasets, indicating a potential need for dataset-specific strategies.

Masking vs. relabeling. Table 6 shows that for Misclassified-*O*, relabeling usually outperforms masking, suggesting model-predicted labels are often correct. For Misclassified-*entity* however, both methods perform similarly, indicating that correcting or simply removing the noise yields the same benefit – consistent with its low relabeling accuracy in Table 5.

Combining categories. Combining the modifications for Learned-*O*, Misclassified-*O* and Misclassified-*entity* performs worse than the best single-category intervention. This is counterintu-

		Token Metrics Score	Related Work Approach	Related Work Score
NoiseBench	Expert	**90.5±0.2**	CL	90.0±0.2
	Crowd++	88.4±0.2	MSR	**88.5±1.1**
	Distant	74.4±0.4	MSR	**75.8±1.4**
	Crowd	**74.3±0.4**	BOND	74.1±0.5
	Weak	67.4±0.4	MSR	**69.5±0.3**
	LLM	64.6±0.4	L2R	**65.3±4.2**
	Ger. Expert	**79.9±0.3**	CL	79.6±0.3
	Ger. LLM	57.9±0.5	MSR	**64.0±0.7**
NoisyNER	NoisyNER 1	36.1±0.3	STGN	**40.0**
	NoisyNER 2	44.1±0.6	STGN	**47.7**
	NoisyNER 3	43.1±1.3	STGN	**44.0**
	NoisyNER 4	45.4±0.7	STGN	**51.8**
	NoisyNER 5	53.1±0.6	STGN	**53.1**
	NoisyNER 6	**55.3±1.3**	STGN	55.0
	NoisyNER 7	**62.5±1.9**	STGN	60.1

Table 7: Comparison of our token metrics approach and related work. The token metrics column shows the best scores from Table 6, while the related work column shows the score from the corresponding best method. **Bolded** are the highest test F1 scores in each row.

itive and may be due to overlap or interference between strategies, which could reduce overall effectiveness.

Comparison to state-of-the-art. Table 7 compares our best scores with previous work. For NoiseBench (English and German), we use results from Merdjanovska et al. (2024), and for NoisyNER, from Wu et al. (2022). State-of-the-art methods include MSR (Zhu et al., 2023a), CL (Northcutt et al., 2021), BOND (Liang et al., 2020), L2R (Ren et al., 2018), and STGN (Wu et al., 2022). In the token metric approach we do not use validation sets, however some other approaches like BOND and MSR require one. In these cases, for NoiseBench the corresponding noisy validation sets were used. For STGN on NoisyNER, they used a 10% validation set, however it is unclear whether its labels were clean or noisy.

Outside of English NoiseBench, which we used for metric selection, we outperform SOTA only for NoisyNER 6 and 7. We also improve over SOTA on the Expert sets, but the gains are not notable.

6 Related work

Detecting and correcting label errors. The annotation error detection paper (Klie et al., 2023) gives an extensive overview of different strategies for detecting label errors. They include sample metrics (scorer methods) in the comparison (Swayamdipta et al., 2020; Larson et al., 2020), but also methods ensembling multiple models (Reiss et al., 2020; Loftsson, 2009; Rodrigues et al., 2013) and other approaches (Northcutt et al., 2021; Amiri et al., 2018) . Their NER evaluation is quite limited and only detection strategies are explored, without considering relabeling.

Noise-robust learning. Cleaning the dataset (Northcutt et al., 2021; Reiss et al., 2020) by removing or relabelling the incorrect annotations is one strategy for learning under label noise. Other options include reweighting the potentially erroneous samples (Wang et al., 2019; Ren et al., 2018), delaying memorization (Zhou and Chen, 2021) and multi-stage pipelines (Zhu et al., 2023a; Liang et al., 2020; Yu et al., 2021; Wang et al., 2022). Recent noise-robust learning studies have analysed the use of validation sets (Zhu et al., 2023b; Li et al., 2025), suggesting that the use of large or clean validation data leads to overly optimistic evaluation.

7 Conclusion

This paper explores how token-level metrics can be used to correct label annotations for NER. We assess if each metric can be used as a separation feature between clean and noisy samples. For this, we look at four token categories separately.

When comparing different metrics, we see an added benefit of using metrics training dynamics-based, as opposed to only well-established estimates like the maximum softmax probability. Also, we found that layer agreement metrics perform worse than the other two types.

We compare two strategies to deal with the erroneous samples: relabeling and masking. With relabeling, we see good improvements in the resulting noise shares. We are able to successfully reduce the noise share across all datasets and variants by 3.3 percentage points on average by either correcting missing mention or non-entity errors. As a next step, we re-trained a model with the modified dataset, to evaluate the impact of this approach on the final test scores. We see that the reduction in noise share translates in modest improvements with retraining for all datasets and noise types apart from Expert noise.

The token metrics approach works for Misclassified-*O* and Misclassified-*entity* and it fails for Learned categories. This shows that once a model has memorized a label error, it is much more challenging to detect. Finally, our experiments span three different languages and show that the metric and threshold selection generalizes well.

Limitations

One of the main limitations of this paper is the fact that the metric selection process uses information from the clean training data. As such, it can not be used to determine the best metric and parameters on a given noisy dataset.

Furthermore, we tried to include a diverse set of candidate metrics, representing different ideas, but there are many more potential sample metrics to explore in the context of NER and token-level selection.

The sample metrics used are designed for classification tasks in general, so it would be interesting to see this analysis for other tasks. This could extend outside of NLP, so in that sense our study is limited to NER, even though a wider analysis of these methods is possible.

Acknowledgments

We thank all reviewers for their valuable comments. Elena Merdjanovska and Alan Akbik are supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2002/1 "Science of Intelligence" – project number 390523135. Alan Akbik is supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Emmy Noether grant "Eidetic Representations of Natural Language" (project number 448414230).

References

Hadi Amiri, Timothy Miller, and Guergana Savova. 2018. Spotting spurious data with neural networks. pages 2006–2016, New Orleans, Louisiana. Association for Computational Linguistics.

Robert Baldock, Hartmut Maennel, and Behnam Neyshabur. 2021. Deep learning through the lens of example difficulty. *Advances in Neural Information Processing Systems*, 34:10876–10889.

Pengfei Chen, Junjie Ye, Guangyong Chen, Jingwei Zhao, and Pheng-Ann Heng. 2020. Beyond class-conditional assumption: A primary attempt to combat instance-dependent label noise. In *AAAI Conference on Artificial Intelligence*.

Pengfei Chen, Junjie Ye, Chen Guangyong, Jingwei Zhao, and Pheng-Ann Heng. 2021. Beyond class-conditional assumption: A primary attempt to combat instance-dependent label noise. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35:11442–11450.

Muyang He, Shuo Yang, Tiejun Huang, and Bo Zhao. 2024. Large-scale dataset pruning with dynamic uncertainty. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 7713–7722.

Michael A Hedderich, Dawei Zhu, and Dietrich Klakow. 2021. Analysing the noise model error for realistic noisy label data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7675–7684.

Chuanyang Hu, Shipeng Yan, Zhitong Gao, and Xuming He. 2024. Mild: Modeling the instance learning dynamics for learning with noisy labels. *Preprint*, arXiv:2306.11560.

Ajay J. Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. 2009. Multi-class active learning for image classification. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2372–2379.

Jan-Christoph Klie, Bonnie Webber, and Iryna Gurevych. 2023. Annotation Error Detection: Analyzing the Past and Present for a More Coherent Future. *Computational Linguistics*, 49(1):157–198.

Stefan Larson, Adrian Cheung, Anish Mahendran, Kevin Leach, and Jonathan K. Kummerfeld. 2020. Inconsistencies in crowdsourced slot-filling annotations: A typology and identification methods. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5035–5046, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Yuepei Li, Kang Zhou, Qiao Qiao, Qing Wang, and Qi Li. 2025. Re-examine distantly supervised NER: A new benchmark and a simple approach. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 10940–10959, Abu Dhabi, UAE. Association for Computational Linguistics.

Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. 2020. Bond: Bert-assisted open-domain named entity recognition with distant supervision. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Hrafn Loftsson. 2009. Correcting a POS-tagged corpus using three complementary methods. In *Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009)*, pages 523–531, Athens, Greece. Association for Computational Linguistics.

Elena Merdjanovska, Ansar Aynetdinov, and Alan Akbik. 2024. NoiseBench: Benchmarking the impact of real label noise on named entity recognition. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18182–18198, Miami, Florida, USA. Association for Computational Linguistics.

Curtis Northcutt, Lu Jiang, and Isaac Chuang. 2021. Confident learning: Estimating uncertainty in dataset labels. *J. Artif. Int. Res.*, 70:1373–1411.

Geoff Pleiss, Tianyi Zhang, Ethan R. Elenberg, and Kilian Q. Weinberger. 2020. Identifying mislabeled data using the area under the margin ranking. In *NeurIPS*.

Frederick Reiss, Hong Xu, Bryan Cutler, Karthik Muthuraman, and Zachary Eichenberger. 2020. Identifying incorrect labels in the conll-2003 corpus. In *Conference on Computational Natural Language Learning*.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. 2018. Learning to Reweight Examples for Robust Deep Learning. In *ICML*.

Filipe Rodrigues, Francisco Pereira, and Bernardete Ribeiro. 2013. Learning from multiple annotators: Distinguishing good from random labelers. *Pattern Recognition Letters*, 34(12):1428–1436.

Susanna Rücker and Alan Akbik. 2023. CleanCoNLL: A Nearly Noise-Free Named Entity Recognition Dataset. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Stefan Schweter and Alan Akbik. 2021. FLERT: Document-Level Features for Named Entity Recognition. *Preprint*, arXiv:2011.06993.

Shoaib Ahmed Siddiqui, Nitarshan Rajkumar, Tegan Maharaj, David Krueger, and Sara Hooker. 2022. Metadata archaeology: Unearthing data subsets by leveraging training dynamics. *Preprint*, arXiv:2209.10015.

Hwanjun Song, Minseok Kim, and Jae-Gil Lee. 2019. SELFIE: Refurbishing unclean samples for robust deep learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5907–5915. PMLR.

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. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9275–9293, Online. Association for Computational Linguistics.

Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J. Gordon. 2019. An empirical study of example forgetting during deep neural network learning. In *International Conference on Learning Representations*.

Haobo Wang, Ruixuan Xiao, Yiwen Dong, Lei Feng, and Junbo Zhao. 2022. ProMix: combating label noise via maximizing clean sample utility. *arXiv preprint arXiv:2207.10276*.

Zihan Wang, Jingbo Shang, Liyuan Liu, Lihao Lu, Jiacheng Liu, and Jiawei Han. 2019. Crossweigh: Training named entity tagger from imperfect annotations. In *Conference on Empirical Methods in Natural Language Processing*.

Tingting Wu, Xiao Ding, Minji Tang, Hao Zhang, Bing Qin, and Ting Liu. 2022. STGN: an implicit regularization method for learning with noisy labels in natural language processing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7587–7598, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yue Yu, Simiao Zuo, Haoming Jiang, Wendi Ren, Tuo Zhao, and Chao Zhang. 2021. Fine-tuning pre-trained language model with weak supervision: A contrastive-regularized self-training approach. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1063–1077.

Urchade Zaratiana, Nadi Tomeh, Pierre Holat, and Thierry Charnois. 2023. GLiNER: Generalist Model for Named Entity Recognition using Bidirectional Transformer. *Preprint*, arXiv:2311.08526.

Wenxuan Zhou and Muhao Chen. 2021. Learning from noisy labels for entity-centric information extraction. *arXiv preprint arXiv:2104.08656*.

Dawei Zhu, Xiaoyu Shen, Michael Hedderich, and Dietrich Klakow. 2023a. Meta Self-Refinement for Robust Learning with Weak Supervision. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1043–1058, Dubrovnik, Croatia. Association for Computational Linguistics.

Dawei Zhu, Xiaoyu Shen, Marius Mosbach, Andreas Stephan, and Dietrich Klakow. 2023b. Weaker than you think: A critical look at weakly supervised learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14229–14253, Toronto, Canada. Association for Computational Linguistics.

Henry Zou and Cornelia Caragea. 2023. JointMatch: A unified approach for diverse and collaborative pseudo-labeling to semi-supervised text classification. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7290–7301, Singapore. Association for Computational Linguistics.

A Calculating the Detection F-score

Let one category consist of n samples at epoch e. For each sample $i \in \{1, 2, \ldots, n\}$, and each epoch $e \in \{1, 2, \ldots, t\}$ let:

$$c_i^e = \begin{cases} 1 & \text{if sample } i \text{ is actually incorrect,} \\ 0 & \text{otherwise} \end{cases}$$

$$d_i^e = \begin{cases} 1 & \text{if sample } i \text{ is detected as noisy,} \\ 0 & \text{otherwise} \end{cases}$$

Detection precision is defined as the proportion of detected noisy samples that are actually noisy:

$$Prec. = \frac{\sum_{i=1}^{n} d_i c_i}{\sum_{i=1}^{n} d_i}$$

Detection recall is defined as the proportion of actually noisy samples that are correctly detected:

$$max\_noisy = \max_{e=1}^{t} \sum_{i=1}^{n} c_i^t$$

$$Rec. = \frac{\sum_{i=1}^{n} d_i c_i}{max\_noisy}$$

Here we used a modified recall value, where instead of using the number of noisy samples in a given category at a given epoch, we use the number of noisy samples from the epoch where this number is maximal. We do this so that the number of samples in the category does not bias the best parameter selection.

For $\beta = 0.5$, the $F_{0.5}$ score is:

$$F_{0.5}^e = \frac{1.25 \cdot Prec. \cdot Rec.}{0.25 \cdot Prec. + Rec.}$$

B Implementation Details

Baseline parameters. We fine-tune an `xlm-roberta-large` model, for 10 epochs, using a batch size of 8 and learning rate of 5.0e-6. For the English and German NoiseBench datasets, we add the document context (Schweter and Akbik, 2021), while for NoisyNER we do not, because the data does not include document boundaries.

When using the layer-wise classifier outputs, we use the same model and same parameters. Additionally, before the main fine-tuning, we tune the decoders with a learning rate of 0.3 and for 10 epochs.

Metric selection. For MSP, prediction-margin, iteration-learned, correctness and confidence the list of thresholds we consider is [0.1, 0.2, 0.3 ... 0.9]. For variability and entropy-history it is [0.05, 0.1, 0.15 ... 0.45]. For cross-entropy, label-entropy and predictive-entropy, we take the interval between the minimal and maximal value and divide it by 10. For the remaining layer agreement metrics and MILD, we use increments of 1 in the interval between the minimal and maximal values. This means that for these metrics we do not always have a list of 10 threshold options, as we did for the others.

metric	score	epoch	threshold
cross-entropy	0.54	1	> 0.3
MSP	0.54	1	< 0.7
prediction-margin	0.53	1	< 0.6
predictive-entropy	0.53	1	> 1.1
confidence	0.59	8	< 0.8
variability	0.63	9	> 0.1
correctness	0.62	9	< 0.9
iteration-learned	0.62	9	> 0.2
MILD	0.62	9	> -8
entropy-history	0.61	8	> 0.1
prediction-depth	0.55	1	> 13
first-layer	0.44	6	> 1
agreement-predicted	0.56	1	< 20
agreement-true	0.56	1	< 20
layer-entropy	0.54	1	> 0.5

Table 8: Best parameter settings for Learned-*entity* tokens

C Best Parameter Sets

In Tables 8 - 11 we show the parameter settings which result in the best average scores across NoiseBench (*average* score in Experiment 2).

Learned-*O* tokens. Table 8 summarizes the results for this type of tokens. Here, confidence estimates and layer-based metrics consistently perform best during earlier training epochs, suggesting that these methods are immediately able to discriminate problematic tokens. For the layer-based metrics this may be attributed to the initialization phase, where the classifier heads are trained before overfitting of the transformers begins. In contrast, training dynamics metrics, which aggregate values over iterations, achieve best performance in later epochs, as expected due to their temporal nature.

Misclassified-*O* tokens. As shown in Table 10, the choice of metric and threshold has minimal impact on detection performance. In fact, for example, for MSP the optimal threshold is the minimum value. This means that simply relabeling all tokens in this category yields $F_{0.5}$ scores of 0.89, which is close to the maximum of 0.91 achieved using confidence as a criterion.

metric	score	epoch	threshold
cross-entropy	0.34	9	> 0.1
MSP	0.34	9	< 0.9
prediction-margin	0.35	9	< 0.9
predictive-entropy	0.35	9	> 0.3
confidence	0.46	9	< 0.7
variability	0.36	9	> 0.2
correctness	0.48	9	< 0.7
iteration-learned	0.47	9	> 0.4
MILD	0.48	9	> -4
entropy-history	0.44	9	> 0.2
prediction-depth	0.32	9	> 7
first-layer	0.26	9	> 3
agreement-predicted	0.31	9	< 20
agreement-true	0.31	9	< 20
layer-entropy	0.30	9	> 0.5

Table 9: Best parameter settings for Learned - *entity* tokens

metric	score	epoch	threshold
cross-entropy	0.89	1	> 0
MSP	0.89	1	> 0
prediction-margin	0.89	1	> 0
predictive-entropy	0.87	2	< 1.9
confidence	0.91	2	< 0.4
variability	0.90	2	< 0.1
correctness	0.88	2	< 0.6
iteration-learned	0.89	1	< 0
MILD	0.89	1	> -2
entropy-history	0.81	3	< 0.3
prediction-depth	0.80	5	< 24
first-layer	0.80	5	> 0
agreement-predicted	0.82	5	> 0
agreement-true	0.82	5	< 24
layer-entropy	0.81	5	< 1.2

Table 10: Best parameter settings for Misclassified - *O* tokens

metric	score	epoch	threshold
cross-entropy	0.68	4	> 1.5
MSP	0.67	4	> 0.3
prediction-margin	0.67	4	> 0
predictive-entropy	0.67	4	< 1.8
confidence	0.68	4	< 0.3
variability	0.68	4	< 0.1
correctness	0.68	4	< 0.3
iteration-learned	0.66	4	< 0.8
MILD	0.68	4	> 1
entropy-history	0.67	4	< 0.4
prediction-depth	0.60	4	< 21
first-layer	0.59	7	> 0
agreement-predicted	0.60	7	> 6
agreement-true	0.62	4	< 6
layer-entropy	0.60	7	< 1.5

Table 11: Best parameter settings for Misclassified - *entity* tokens

For this category, optimal detection of noisy tokens generally occurs early in training for both training dynamics metrics and confidence-based estimates. This suggests that the pretrained model already possesses sufficient generalization capabilities to flag these errors without requiring extensive fine-tuning. Interestingly, layer metrics achieve their best performance at epoch 5, slightly later than other metric types.

Learned-*entity* tokens. The best detection performance is at epoch 9, as shown in Table 9.

Misclassified-*entity* Table 11 shows the results for this category. Epoch 4 consistently yields the best performance across all metric types, indicating a highly localized signal in the training process. As with Misclassified-*O*, even a naive strategy, such as relabeling all tokens where BvSB > 0, achieves an $F_{0.5}$ of 0.67, which is close to the category's best value of 0.68.

D Training Layer-Wise Classifiers (LWC)

To calculate the layer agreement metrics, we need the model's predictions at the end of each transformer block. For this, we add a classifier layer at the end of each block. This also requires a decoder initialization phase, where we first fine-tune only the linear classifier layers, while the transformers are static. After this phase, we jointly fine-tune all layers.