Detecting Label Errors in Token Classification Data

Wei-Chen Wang
wangeric@mit.edu
Cleanlab, MIT

Jonas Mueller
jonas@cleanlab.ai
Cleanlab

Abstract

Mislabeled examples are a common issue in real-world data, particularly for tasks like token classification where many labels must be chosen on a fine-grained basis. Here we consider the task of finding sentences that contain label errors in token classification datasets. We study 11 different straightforward methods that score tokens/sentences based on the predicted class probabilities output by a (any) token classification model (trained via any procedure). In precision-recall evaluations based on real-world label errors in entity recognition data from CoNLL-2003, we identify a simple and effective method that consistently detects those sentences containing label errors when applied with different token classification models.

1 Introduction

It has recently come to light that many supervised learning datasets contain numerous incorrectly labeled examples [13]. To efficiently improve the quality of such data, Label Error Detection (LED) has emerged as a task of interest [10, 14, 11], in which algorithms flag examples whose labels are likely wrong for reviewers to inspect/correct. This may be done by means of a score for each example which reflects its estimated *label quality*. A useful score ranks mislabeled examples higher than others, allowing reviewers to much more efficiently identify the label errors in a dataset.

This paper considers LED for token classification tasks (such as *entity recognition*) in which each token in a sentence has been given its own class label. While it is possible to score the labels of individual tokens, reviewing a candidate token flagged as potentially mislabeled requires looking at the entire sentence to understand the broader context. Here we propose `worst-token`, a method to score sentences based on the likelihood that they contain some mislabeled tokens, such that sentences can be effectively ranked for efficient label review[1]. We evaluate the LED performance of this approach and others on real-world data with naturally occuring label errors, unlike many past LED evaluations based on synthetically-introduced label errors [1, 11, 7], for which conclusions may differ from real-world errors [10, 8, 23].

Related Work. Extensive research has been conducted on standard classification with noisy labels [19, 23, 2, 11, 12, 1]. Our work builds on label quality scoring methods for classification data studied by Northcutt et al. [14], Kuan and Mueller [10], which merely depend on predictions from a trained multiclass classification model. These methods are straightforward to implement and broadly applicable, being compatible with any classifier and training procedure, as is our `worst-token` method.

Only a few prior works have studied label error detection for token classification tasks specifically [22, 17, 9]. Also aiming to score the overall label quality of an entire sentence like us, Wang et al. [22] propose `CrossWeigh` which: trains a large ensemble of token classification models with

[1] Code to run our method: https://github.com/cleanlab/cleanlab
Code to reproduce our results: https://github.com/cleanlab/token-label-error-benchmarks

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

entity-disjoint cross-validation, and scores a sentence based on the number of ensemble-member class predictions that deviate from the given label, across all tokens in the sentence. Reiss et al. [17] propose a similar approach to LED in token classification, which also relies on counting deviations between token labels and corresponding class predictions output by a large ensemble of diverse models trained via cross-validation. Unlike our `worst-token` method, the methods of Wang et al. [22], Reiss et al. [17] are more computationally complex due to ensembling many models, and do not account for the confidence of individual predictions (as they are based on hard class predictions rather than estimated class probabilities). Given the LED performance of `worst-token` improves with a more accurate token classifier, we expect `worst-token` to benefit from ensembling's reliable predictive accuracy improvement in the same way that ensembling benefits the methods of Wang et al. [22], Reiss et al. [17]. We also tried entity-disjoint vs. standard data splitting, but did not observe benefits from the former variant (it often removes a significant number of entities).

Klie et al. [9] study various approaches for LED in token classification, but only consider scores for individual tokens rather than entire sentences. Here we compare against some of the methods that performed best in their study. The studies of Klie et al. [9], Northcutt et al. [14], Kuan and Mueller [10] indicate that label errors can be more effectively detected by considering the confidence level of classifiers rather than only their hard class predictions. Instead depending on predicted class probabilities for each token, our `worst-token` method appropriately accounts for classifier confidence.

2 Methods

Typical token classification data is composed of many sentences (i.e. training instances), each of which is split into individual tokens (words or sub-words) where each token is labeled as one of K classes (i.e. entities in entity recognition). Given a sentence x, a trained token classification model $M(\cdot)$ outputs predicted probabilities $\mathbf{p} = M(x)$ where p_{ij} is the probability that the ith token in sentence x belongs to class j. Throughout, we assume these probabilities are *out-of-sample*, computed from a copy of the model that did not see x during training (e.g. because x is in test set, or cross-validation was utilized).

Using p, we first consider evaluating the individual per-token labels. Here we apply effective LED methods for standard classification settings [14] by simply treating each token as a separate independent instance (ignoring which sentence it belongs to). Following Kuan and Mueller [10], we compute a label quality score $q_i \in [0, 1]$ for the ith token (assume it is labeled as class k) via one of the following options:

- self-confidence (`sc`): $q_i = p_{ik}$, i.e. predicted probability of the given label for this token.
- normalized margin (`nm`): $q_i = p_{ik} - p_{i\tilde{k}}$ with $\tilde{k} = \text{argmax}_j \{p_{ij}\}$
- confidence-weighted entropy (`cwe`): $q_i = \dfrac{p_{ik}}{H(p_i)}$ where $H(p_i) = -\dfrac{1}{\log K} \sum_{j=1}^{K} p_{ij} \log(p_{ij})$

Higher values of these label quality scores correspond to tokens whose label is more likely to be correct [10]. We can alternatively evaluate the per-token labels via the Confident Learning algorithm of Northcutt et al. [14], which classifies each token as correclty labeled or not ($b_i = 1$ if this token is flagged as likely mislabeled, = 0 otherwise) based on adaptive thresholds set with respect to per-class classifier confidence levels.

For one sentence with n word-level tokens, we thus have:

- $\mathbf{p}$, a $n \times K$ matrix where p_{ij} is model predicted probability that the ith token belongs to class j.
- $\mathbf{l} = [l_1, \ldots, l_n]$, where $l_i \in \{0, \ldots, K-1\}$ is the given class label of the ith token.
- $\mathbf{q} = [q_1, \ldots, q_n]$, where q_i is a label quality score for the ith token (one of the above options).
- $\mathbf{b} = [b_1, \ldots, b_n]$, where $b_i = 1$ if ith token is flagged as potentially mislabeled, otherwise $b_i = 0$.

Recall that to properly verify whether a token is really mislabeled, a reviewer must read the full sentence containing this token to understand the broader context. Thus the most efficient way to review labels in a dataset is to prioritize inspection of those *sentences most likely to contain a*

mislabeled token. We consider 11 methods to estimate an overall quality score $s(x)$ for the sentence x, where higher values correspond to sentences whose labels are more likely all correct.

1. `predicted-difference`: The number of disagreements between the given and model-predicted class labels over the tokens in the sentence, also utilized in the methods of Wang et al. [22], Reiss et al. [17]. Here we break sentence-score ties in favor of the highest-confidence disagreement. More formally:
$$s(x) = -|\mathcal{R}| - \max_{i \in \mathcal{R}} p_{i,\widehat{l}_i}$$
where $\widehat{l}_i = \text{argmax}_j\{p_{ij}\}$ and $\mathcal{R} = \{i : \widehat{l}_i \neq l_i\}$. If $\mathcal{R} = \varnothing$, we let $\max_{i \in \mathcal{R}} p_{i,\widehat{l}_i} = 0$.
2. `bad-token-counts`: $s(x) = -\sum_i b_i$, the number of Confident Learning flagged tokens. Similarly considered by Klie et al. [9], this approach is a natural token-classification extension of the method of Northcutt et al. [14] for LED in standard classification tasks.
3. `bad-token-counts-avg`: Again scoring based on number of tokens flagged as potentially mislabeled, but now breaking ties primarily via the average label quality score of the flagged tokens and secondarily via the average label quality score of the other tokens. More formally:
$$s(x) = -\sum_i b_i + \frac{1}{|\mathcal{R}|}\sum_{i \in \mathcal{R}} q_i + \frac{\epsilon}{|\mathcal{S}|}\sum_{i \in \mathcal{S}} q_i$$
where $\mathcal{R} = \{i : b_i = 1\}$, $\mathcal{S} = \{i : b_i = 0\}$, and ϵ is some small constant.
4. `bad-token-counts-min`: Similar to `bad-token-counts-avg`, but break ties using minimum token quality rather than average token quality. More formally:
$$s(x) = -\sum_i b_i + \min_{i \in \mathcal{R}} q_i + \epsilon \cdot \min_{i \in \mathcal{S}} q_i$$
5. `good-fraction`: Fraction of tokens not flagged as potential issues, $s(x) = -\frac{1}{n}\sum_{i=1}^{n} b_i$.
6. `penalize-bad-tokens`: Penalize flagged tokens based on their corresponding label quality scores. More formally,
$$s(x) = 1 - \frac{1}{n}\sum_{i=1}^{n} b_i(1 - q_i)$$
7. `average-quality`: Average label quality of tokens in the sentence, $s(x) = \frac{1}{n}\sum_{i=1}^{n} q_i$.
8. `product`: $s(x) = \sum_i \log(q_i + c)$, where c is a constant hyperparameter. This score places greater emphasis on tokens with low estimated label-quality, while still being influenced by all tokens' quality (like the previous `average-quality` method). With q based on `sc` or `nm` token-scores, the `product` and `average-quality` methods are natural sentence extensions of the CU or PM methods considered in Klie et al. [9] for token-level LED.
9. `expected-bad`: A rough approximation of the expected number of mislabeled tokens in sentence. More formally:
$$s(x) = \sum_{j=1}^{\min(n,J)} j \cdot q^{(j)}$$
where $q^{(i)}$ is the ith lowest token label quality score in this sentence, and J is a hyperparameter. If using the `sc` label-quality score, $1 - q^{(i)}$ can be considered a loose proxy for the probability of having at least i label errors in this sentence.
10. `expected-alt`: Similar to `expected-bad`, but only considering the likelihood of any label error rather than how many might be in this sentence. More formally:
$$s(x) = \sum_{j=1}^{\min(n,J)} q^{(j)}$$
11. `worst-token`: The quality of the worst-labeled token in the sentence determines its overall quality score, $s(x) = \min\{q_1, q_2, \ldots, q_n\}$. This is a reasonable way to rank the sentences most likely to have some label error, i.e. those most worthy of manual review.

3 Results

For evaluation, we apply each sentence scoring method to the given class labels in the CoNLL-2003 named entity recognition dataset [21]. We restrict our attention to the test set, for which all ground truth label errors were identified by Wang et al. [22] (see Appendix B). We consider two different models to produce per-token predicted probabilities: `bert` [6] and `xlm` [3], and we consider a second variant of the dataset (`unmerged`) with more classes based on additional consideration of `B-` and `I-` entity-prefixes. See Appendix A for all details.

Recall the sentence scores $s(x)$ are used to prioritize which sentences most likely contain label errors. Thus we consider evaluation metrics from information retrieval, which depend on the ranking of sentences induced by $s(x)$ rather than the magnitude of its values. Sentences that contain any mislabeled token are considered true positives when we compute metrics like: **AUROC** (and **AUPRC**) for area under the receiver operating characteristic (and precision-recall) curve. Our third metric, **Lift @ #Errors**, measures how many times more prevalent labels errors are within the top-T scoring sentences vs. all sentences. Here T is the number of true positives ($T = 184$ for `bert` and `xlm`, $T = 186$ for `bert-unmerged`). The Lift metric favors high-precision scores, while AUROC and AUPRC consider both precision and recall, favoring scores capable of detecting a meaningful fraction of all true positives. AUPRC is sometimes preferred over AUROC in settings where true positives are rare [5, 18].

Table 1 presents some results and others are in Appendix D. Our results show that `worst-token` (using the `sc` token-score) generally achieves the best LED performance across the three experiments. To most usefully rank sentences for identifying label errors, one should thus account for classifier confidence but not be directly influenced by all tokens' estimated quality (which may be noisy).

Table 1: AUPRC achieved by different sentence and token scoring methods.

Sentence Score	Token Score	bert	xlm	bert-unmerged
`predicted-difference`		0.3422	0.3412	0.3190
`bad-token-counts`		0.3087	0.3186	0.3291
`bad-token-counts-avg`	`sc`	0.3740	0.3697	0.3768
	`nm`	0.3702	0.3603	0.3740
	`cwe`	0.3597	0.3597	0.3609
`bad-token-counts-min`	`sc`	0.3804	0.3759	0.3901
	`nm`	0.3744	0.3662	0.3822
	`cwe`	0.3695	0.3602	0.3607
`good-fraction`		0.3131	0.3159	0.2996
`average-quality`	`sc`	0.3022	0.3349	0.2574
	`nm`	0.3066	0.3143	0.2648
	`cwe`	0.2767	0.3495	0.2572
`penalize-bad-tokens`	`sc`	0.3423	0.3321	0.3229
	`nm`	0.3368	0.3126	0.3221
	`cwe`	0.3191	0.3380	0.3023
`product`	`sc`	0.3794	0.3559	0.3726
	`nm`	0.3807	0.3533	0.3823
	`cwe`	0.3519	0.3783	0.3359
`expected-bad`	`sc`	0.3383	0.3485	0.3532
	`nm`	0.3776	0.3227	0.3513
	`cwe`	0.3191	0.3541	0.2980
`expected-alt`	`sc`	0.3927	0.3628	0.3614
	`nm`	0.3850	0.3342	0.3603
	`cwe`	0.3335	0.3620	0.3114
`worst-token`	`sc`	**0.4357**	**0.4021**	**0.4236**
	`nm`	0.4243	0.3963	0.3933
	`cwe`	0.3215	0.3815	0.2974

References

[1] C. E. Brodley and M. A. Friedl. Identifying mislabeled training data. *Journal of Artificial Intelligence Research*, 11:131–167, 1999.

[2] P. Chen, B. B. Liao, G. Chen, and S. Zhang. Understanding and utilizing deep neural networks trained with noisy labels. In *International Conference on Machine Learning*, 2019.

[3] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised cross-lingual representation learning at scale. In *ACL*, 2020.

[4] L. David S. bert-base-ner. *Hugging Face Models*. URL https://huggingface.co/dslim/bert-base-NER.

[5] J. Davis and M. Goadrich. The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240, 2006.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

[7] K. Gu, X. Masotto, V. Bachani, B. Lakshminarayanan, J. Nikodem, and D. Yin. An instance-dependent simulation framework for learning with label noise. *arXiv preprint arXiv:2107.11413*, 2021.

[8] L. Jiang, D. Huang, M. Liu, and W. Yang. Beyond synthetic noise: Deep learning on controlled noisy labels. In *International Conference on Machine Learning*, 2020.

[9] J.-C. Klie, B. Webber, and I. Gurevych. Annotation error detection: Analyzing the past and present for a more coherent future. *arXiv preprint arXiv:2206.02280*, 2022.

[10] J. Kuan and J. Mueller. Model-agnostic label quality scoring to detect real-world label errors. In *ICML DataPerf Workshop*, 2022.

[11] N. M. Müller and K. Markert. Identifying mislabeled instances in classification datasets. In *International Joint Conference on Neural Networks*, 2019.

[12] N. Natarajan, I. S. Dhillon, P. K. Ravikumar, and A. Tewari. Learning with noisy labels. In *Advances in Neural Information Processing Systems*, 2013.

[13] C. G. Northcutt, A. Athalye, and J. Mueller. Pervasive label errors in test sets destabilize machine learning benchmarks. In *Proceedings of the 35th Conference on Neural Information Processing Systems Track on Datasets and Benchmarks*, December 2021.

[14] C. G. Northcutt, L. Jiang, and I. L. Chuang. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70:1373–1411, 2021.

[15] L. A. Ramshaw and M. P. Marcus. Text chunking using transformation-based learning. In *Natural language processing using very large corpora*, pages 157–176. Springer, 1999.

[16] A. Ratnaparkhi. *Maximum entropy models for natural language ambiguity resolution*. University of Pennsylvania, 1998.

[17] F. Reiss, H. Xu, B. Cutler, K. Muthuraman, and Z. Eichenberger. Identifying incorrect labels in the CoNLL-2003 corpus. In *Proceedings of the 24th conference on computational natural language learning*, pages 215–226, 2020.

[18] T. Saito and M. Rehmsmeier. The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PloS one*, 10(3), 2015.

[19] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[20] The team at Hugging Face. xlm-roberta-large-finetuned-conll03-english. *Hugging Face Models*. URL https://huggingface.co/xlm-roberta-large-finetuned-conll03-english.

[21] E. F. Tjong Kim Sang and F. De Meulder. 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.

[22] Z. Wang, J. Shang, L. Liu, L. Lu, J. Liu, and J. Han. Crossweigh: Training named entity tagger from imperfect annotations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[23] J. Wei, Z. Zhu, H. Cheng, T. Liu, G. Niu, and Y. Liu. Learning with noisy labels revisited: A study using real-world human annotations. In *International Conference on Learning Representations*, 2022.

Appendix: Detecting Label Errors in Token Classification Data

A Experiment Details

Dataset. The CoNLL-2003 dataset contains 4 types of named entities: `PER` for persons, `ORG` for organizations, `LOC` for locations, `MISC` for miscellaneous other entities, with `O` being reserved as a label for other types of words that are not named entities. The dataset is in IOB2 format [16], such that all named entities possess an extra `B-` or `I-` prefix, which indicates whether this token is the Beginning of an entity or an Intermediate part of one. We consider this set of 9 classes in `bert` and `xlm`, in which differently prefixed entities are treated as different classes. In the alternative `merged` setting, we consider the `merged` case in where the prefixes are disregarded. In either setting, a sentence is considered "mislabeled" if at least one label of the word-level token differs from original dataset.

Models. For producing predicted probabilities to input into our label quality scoring methods, all models were only trained on the training set of CoNLL-2003, and we use them to produce held-out predictions for the test set. No model has access to the test set examples during training, nor any ground-truth label errors at any point in our evaluation.

We consider three different settings in our experiments, described below in the corresponding order: `bert-unmerged`, `bert`, `xlm`. The first model we consider is a pre-trained `bert-base-NER` Transformer network, which has been found to be an effective token classifier for CoNLL-2003 including the `B-` and `I-` entity-prefixes as additional classes (for a total of 9 classes) [4]. We conduct two additional experiments to verify whether the label error detection results with this model remain consistent in other settings. First, we omit the `B-` and `I-` prefixes such that we only focus on more severe error types, such as `LOC` vs. `ORG` rather than `B-LOC` vs. `I-LOC` label substitutions (which may be of less interest). With this reduced set of entities, we still use the same Bert network, which now predicts amongst a fewer set of 5 classes for each token. Finally, to examine how our methods work when applied with different models, we also obtain a different set of model-predicted probabilities using another pre-trained XLM network: `xlm-roberta-large-finetuned-conll03-english` [20]. Given that the model outputs predictions in the IOB format [15], we again consider the reduced set of 5 classes.

Data Processing. Before applying our label error detection methods, we first preprocess the original CoNLL-2003 dataset. Sentences less than or equal to 1 character, and sentences containing `"#"` are excluded. The latter because `"#"` is a special character reserved by many token classification models to represent subword tokens. We construct each sentence by joining the tokens separated by space, and perform some minor cleanup to ensure that the sentence follows writing conventions (such as no space before comma, and no space after open parenthesis). We convert all-caps tokens into lowercase except for the first character (i.e. `JAPAN -> Japan`), because token classification models tend to partition all-caps tokens into multiple subword-level tokens. This can sometimes result in some undesirable behaviors, such as converting `USA` to `Usa`, or `NBA` to `Nba`. We believe that the benefits outweigh the costs for CoNLL-2003, due to the prevalence of article headlines in which most, if not all, tokens in the sentence are all-caps.

Next, we use a pre-trained token classification model to obtain the model-predicted probabilities. Modern Transformer models first tokenize the sentence into multiple subword-level tokens, possibly different from the given tokens individually labeled in the original dataset. These "subword-level" tokens are typically smaller units than word-level tokens, and the trained model outputs a probability distribution over possible classes for each such token.

Here we reduce these probabilities from subword-level to word-level tokens, so that we can evaluate the given labels as outlined in the main text. Consider the sentence:

```
Minnesota Timberwolves (MIN)
```

where the given tokens from original dataset are [‘‘Minnesota’’, ‘‘Timberwolves’’, ‘‘(’’, ‘‘MIN’’, ‘‘)’’], and the model tokenizes the sentence into subword-level tokens: [‘‘Minnesota’’, ‘‘Timber’’, ‘‘wolves’’, ‘‘(MIN)’’][2]. Let $\mathbf{p}^{(i)} = [p_1^{(i)}, p_2^{(i)}, \ldots, p_K^{(i)}]$ denote the model-predicted probabilities of the ith subword-level token, where K is the number of possible classes. To obtain the predicted probability distribution for ‘‘Timberwolves’’ (used to evaluate its given label in the original dataset), we take the average of the (model-estimated) probabilities for ‘‘Timber’’ $\mathbf{p}^{(2)}$, and ‘‘wolves’’ $\mathbf{p}^{(3)}$. We assign the probability for ‘‘(MIN)’’ $\mathbf{p}^{(4)}$ directly to ‘‘(’’, ‘‘MIN’’ and ‘‘)’’. Similar strategies are applied for all other model-tokenization-induced discrepancies in the data.

We also experimented with alternative probability-pooling methods for adjacent subword-level tokens, such as: a weighted average (with weights proportional to the number of characters in each subword-level token, such that longer strings receive a higher weight), or considering only the predicted probabilities for first subword-level token. Both of these variants produced little differences in their LED performance from the average-pooling technique we employ for the results shown in this paper.

Hyperparameter Settings. 5 of the sentence scoring methods considered in this paper contain a hyperparameter (but not our proposed `worst-token` method). To ensure these other methods are not disadvantaged vs. `worst-token`, we tried many different values of their hyperparameter and report results for the best hyperparameter value. See Appendix C for descriptions of some additional methods not introduced in the main text. The values considered for each method are as follows, with the selected hyperparameter in **bold**.

- `product`: $10^{-1}, 10^{-1.5}, 10^{-2}, 10^{-2.5}, \mathbf{10^{-3}}$
- `worst-token-min-alt`: $0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, \mathbf{0.1}$
- `worst-token-softmin`: $10^{-1}, 10^{-1.1}, 10^{-1.2}, 10^{-1.3}, 10^{-1.4}, \mathbf{10^{-1.5}}$
- `expected-bad`: $\mathbf{2}, 3, 4$
- `expected-alt`: $\mathbf{2}, 3, 4$

Note that `expected-bad` and `expected-alt` with a hyperparameter value of 1 are identical to `worst-token`, so this value is excluded from our hyperparameter sweep.

B Ground Truth Label Errors in CoNLL (via CoNLL++)

Wang et al. [22] manually corrected the entire test set of CoNLL-2003, discovering 186 sentences (5.38% of the original data) that contain at least one token label error. Through extensive manual inspection, we comprehensively verified that basically all CoNLL-2003 label errors (in test set) have been fixed by Wang et al. [22], and their proposed corrections are reliable (we did not find false positives or negatives). Wang et al. [22] named their corrected version `CoNLL++`, which we consider as a source of ground truth to validate candidate label errors for this study.

Reiss et al. [17] also proposed a corrected set of labels for CoNLL-2003, identified via semi-supervised algorithms and limited manual label verification. However, close examination of their proposed corrections reveals that many of them are fundamentally incorrect and many real CoNLL label errors (found by Wang et al. [22]) were not properly identified/fixed by Reiss et al. [17]. For example, consider the first sentence from the test set:

```
Soccer - Japan get lucky win, China in surprise defeat.
```

In the original CoNLL-2003 dataset, `Japan` is labeled `LOC`, and `China` is labeled `PER` (this happens to be a label error, it should be `LOC` instead). Here, we omit the `B-` and `I-` prefix. In the dataset corrected by Reiss et al. [17], `Japan` and `China` are labeled `ORG`, both of which are incorrect. In addition, consider another sentence from the test set:

```
...is not for a stronger dollar either," said Sumitomo's Note.
```

In the dataset corrected by Reiss et al. [17], ‘‘Note’’ is labeled as `PER`, while the correct label should be `O`. The token is also labeled incorrectly in the original dataset, but is corrected by Wang et al. [22].

[2]Different models may result in different tokenization.

Overall comparing against the high-quality (comprehensively manually verified) label corrections in CONLL++, we found the (mostly algorithmically) CoNLL-corrected dataset from Reiss et al. [17] contains many corrections which are not actually valid (we estimate around 8% of their proposed corrections are wrong). Hence, we opted not to base any evaluations on this dataset from Reiss et al. [17]. This additionally highlights the difficulty of algorithmically correcting an entire dataset's labels, which is why we focus on label error detection in this work, developing methods that enable human reviewers to quickly find and fix the label errors.

Here, we present the estimated label noise matrix between the original CoNLL-2003 dataset vs. CoNLL++ on the token-level. The ith row and jth column of the table below represents the percentage of tokens that are labeled i in CoNLL++, and mislabeled as j in the original CoNLL-2003 dataset.

	O	B-MISC	I-MISC	B-PER	I-PER	B-ORG	I-ORG	B-LOC	I-LOC
O	-	0.01%	0.02%	0.01%		0.01%			
B-MISC	5.39%	-				0.14%		2.49%	
I-MISC	18.90%	1.57%	-						
B-PER	0.49%	0.12%		-		0.06%		0.19%	
I-PER	0.17%			0.43%	-				0.09%
B-ORG	0.82%	1.17%		0.18%		-		1.87%	
I-ORG	3.18%		0.68%		0.34%	0.34%	-	0.23%	0.80%
B-LOC	0.91%	0.49%		0.18%		0.43%	0.06%	-	
I-LOC	2.70%		0.77%				0.39%		-

C Variants of our `worst-token` method

For completeness, we also study some minor variants of our proposed methodology. Using the notation in Section 2, we consider the following alternative methods to produce a sentence score $s(x)$ for sentence x:

- `worst-token-min-alt`: We add quality-score penalty d for tokens flagged as likely label errors by Confident Learning [14], and then consider the worst token based on the penalized quality scores. More formally:
$$s(x) = \min_i (q_i + d \cdot b_i)$$
Recall i ranges over the tokens in sentence x. We experimented with different values of constant hyperparameter d. When $d \geqslant 1$, sentences that contain at least one token flagged as a likely label error by Confident Learning are completely separated from the remaining sentences. In other words, if a sentence does not include any flagged tokens, it is guaranteed to rank below all of the sentences with at least one flagged token.
- `worst-token-softmin`: Instead of considering only the worst token's quality score, we softly consider all other token's quality-score (to a lesser degree) as well in the overall sentence score. More formally:
$$s(x) = \langle \mathbf{q}, \text{softmax}_t(\mathbf{1} - \mathbf{q}) \rangle$$
Here $\langle \cdot, \cdot \rangle$ denotes the inner dot product. The temperature of the softmax t is a constant hyperparameter we tried different values for. Settings of t closer to 0 make this approach converge to our `worst-token` method, whereas this approach converges to the `average-quality` method with large values of t.

We evaluate the Lift, AUROC and AUPRC of our proposed method and its variants for comparison. The setup is the same as before, where we report results from the variants with the best hyperparameter settings we could find (to ensure they are not unfairly disadvantaged against `worst-token`). These settings were $d = 0.1$ for `worst-token-min-alt` and $t = 10^{-1.5}$ for `worst-token-softmin`. The best performing scoring method is highlighted in **bold**.

The `worst-token-softmin` variant provides a "second chance" for other mislabeled tokens' (presumably) low quality scores (e.g. the second lowest) to be considered in the overall sentence quality score. For example, consider two sentences with token quality scores $[0.01, 0.99]$ and $[0.011, 0.02]$, respectively. `worst-token` will assign a lower score to the first sentence, but if the first token

Table 2: Lift @ #Errors for our proposed method `worst-token` and its variants.

Sentence score	Token Score	bert	xlm	bert-unmerged
`worst-token`	`sc`	9.02	8.71	8.83
	`nm`	9.02	8.71	8.73
	`cwe`	7.40	7.90	6.35
`worst-token-min-alt`	`sc`	9.02	8.71	8.63
	`nm`	9.02	8.71	8.73
	`cwe`	7.90	7.70	7.83
`worst-token-softmin`	`sc`	8.92	**8.82**	8.53
	`nm`	**9.83**	8.51	**9.12**
	`cwe`	7.50	8.00	6.35

Table 3: AUPRC for our proposed method `worst-token` and its variants.

Sentence score	Token Score	bert	xlm	bert-unmerged
`worst-token`	`sc`	0.4357	0.4021	**0.4236**
	`nm`	0.4243	0.3963	0.3933
	`cwe`	0.3215	0.3815	0.2974
`worst-token-min-alt`	`sc`	**0.4408**	0.4019	0.4181
	`nm`	0.4238	0.3965	0.3945
	`cwe`	0.3643	0.3819	0.3081
`worst-token-softmin`	`sc`	0.4402	**0.4063**	0.4171
	`nm`	0.4388	0.3980	0.4065
	`cwe`	0.3185	0.3799	0.2937

Table 4: AUROC for our proposed method `worst-token` and its variants.

Sentence score	Token Score	bert	xlm	bert-unmerged
`worst-token`	`sc`	0.9058	**0.9141**	**0.8905**
	`nm`	0.9059	0.9134	0.8852
	`cwe`	0.8996	0.9121	0.8834
`worst-token-min-alt`	`sc`	**0.9066**	0.9140	0.8897
	`nm`	0.9058	0.9135	0.8861
	`cwe`	0.9027	0.9119	0.8872
`worst-token-softmin`	`sc`	0.9026	0.9074	0.8878
	`nm`	0.9040	0.9064	0.8830
	`cwe`	0.8962	0.9046	0.8808

turns out not to be a label error, the sentence will become a false positive. On the other hand, `worst-token-softmin` also considers the second lowest token quality score, and assigns a lower score for the second sentence. In this case, it may be more likely for at least one of the tokens in the second sentence to be mislabeled vs. only the first token in the former sentence. Hence `worst-token-softmin` results in attains better Lift @ #Errors than `worst-token`. However being more dependent on all tokens' label quality scores makes `worst-token-softmin` more sensitive to estimation error than `worst-token`. Thus `worst-token-softmin` does not necessarily produce a better overall ranking of the sentences in terms of AUPRC/AUROC.

`worst-token-min-alt` utilizes additional information beyond `worst-token`, which is estimated via Confident Learning for flagging likely label errors. Tables 3 and 4 show that the additions in `worst-token-min-alt` and `worst-token-softmin` can sometimes provide slight benefits to `worst-token`, but we do not find the performance gains to be significant enough to warrant the additional complexity (and hyperparameters) of these variants.

D Additional Results

Tables 5 and 6 provide additional results for the Lift and AUROC metrics. Note the token score field is left empty for sentence scoring methods that do not rely on token scores.

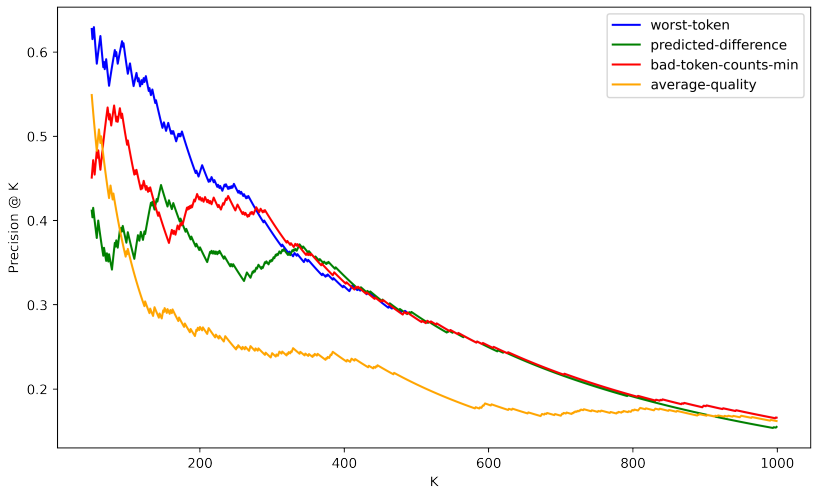
Table 5: Lift @ #Errors for varying sentence and token scoring methods.

Sentence score	Token Score	bert	xlm	bert-unmerged
`predicted-difference`		7.19	6.59	7.14
`bad-token-counts`		7.30	6.89	8.03
`bad-token-counts-avg`	`sc`	7.80	7.09	8.13
	`nm`	7.60	7.40	8.13
	`cwe`	7.30	6.89	7.24
`bad-token-counts-min`	`sc`	7.80	7.09	8.23
	`nm`	7.60	7.40	8.13
	`cwe`	7.30	6.89	7.24
`good-fraction`		5.88	6.18	5.45
`penalize-bad-tokens`	`sc`	5.98	6.18	5.55
	`nm`	5.98	6.18	5.65
	`cwe`	6.38	6.08	5.75
`average-quality`	`sc`	5.07	5.88	4.76
	`nm`	5.07	5.98	4.66
	`cwe`	5.37	6.18	5.26
`product`	`sc`	7.30	6.99	7.24
	`nm`	7.60	7.40	7.64
	`cwe`	6.79	7.09	7.24
`expected-bad`	`sc`	6.99	6.48	6.84
	`nm`	6.69	6.48	7.24
	`cwe`	6.59	6.08	7.34
`expected-alt`	`sc`	6.99	6.48	6.94
	`nm`	6.89	6.59	7.24
	`cwe`	7.09	6.89	7.34
`worst-token`	`sc`	**9.02**	**8.71**	**8.83**
	`nm`	9.02	8.71	8.73
	`cwe`	7.40	7.90	6.35

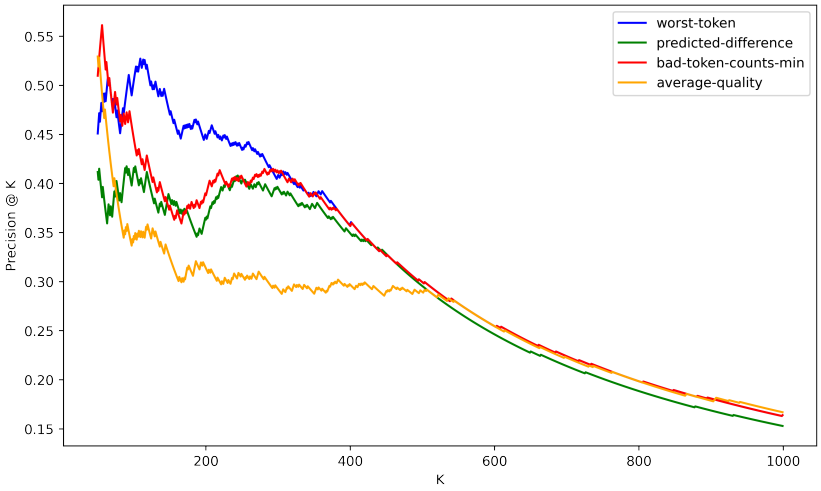
Table 6: AUROC for varying sentence and token scoring methods.

Sentence score	Token Score	bert	xlm	bert-unmerged
predicted-difference		0.8639	0.8633	0.8559
bad-token-counts		0.8188	0.8421	0.8114
bad-token-counts-avg	sc	0.8934	0.9033	0.8718
	nm	0.8932	0.9030	0.8702
	cwe	0.8902	0.9012	0.8725
bad-token-counts-min	sc	0.9026	0.9106	0.8885
	nm	0.9025	0.9102	0.8854
	cwe	0.9023	0.9091	0.8892
good-fraction		0.8147	0.8393	0.8049
penalize-bad-tokens	sc	0.8151	0.8396	0.8053
	nm	0.8151	0.8396	0.8058
	cwe	0.8162	0.8401	0.8064
average-quality	sc	0.8553	0.8895	0.8079
	nm	0.8560	0.8894	0.8068
	cwe	0.8578	0.8913	0.8305
product	sc	0.8900	0.8647	0.8811
	nm	0.8905	0.8646	0.8784
	cwe	0.8870	0.8683	0.8783
expected-bad	sc	0.8946	0.9026	0.8724
	nm	0.8948	0.9017	0.8705
	cwe	0.8922	0.9033	0.8778
expected-alt	sc	0.8963	0.9060	0.8776
	nm	0.8972	0.9044	0.8759
	cwe	0.8950	0.9048	0.8825
worst-token	sc	0.9058	**0.9141**	**0.8905**
	nm	**0.9059**	0.9134	0.8852
	cwe	0.8996	0.9121	0.8834

(a) bert



(b) `xlm`



(c) `bert-unmerged`

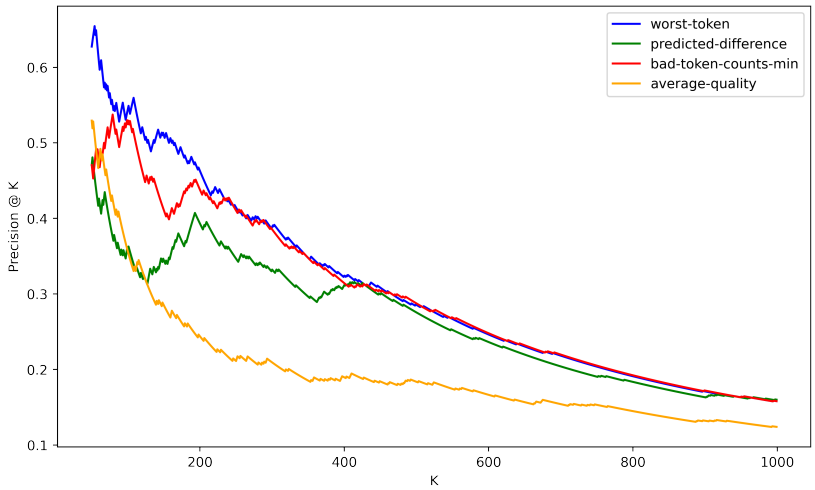


Figure 1: Precision @ K for detecting sentences that contain a label error via our `worst-token` sentence scoring method and three others (over different values of K). Across the 3 experiments, `worst-token` consistently detects sentences containing label errors with higher overall precision.

E Top-ranked Sentences under our `worst-token` method

Based on the given labels and predicted class probabilities from a trained token classification model, our `worst-token` method can quickly help us identify problematic labels in any token classification dataset. Here we list the top 10 sentences with the lowest quality score in the CoNLL-2003 corpus, i.e. those sentences most likely to contain label errors according to our estimates. For tokens estimated to be most problematic, we display their given labels as well as the predicted label output by our trained `bert` token classifier. Because the `O` ("other type of word that is not a named entity") and `MISC` ("miscellaneous entity") labels are often ambiguous, they are excluded from consideration here, as are the `B-` and `I-` prefixes (see Appendix A).

1. `Little change from today's weather expected.`
   Given label: `PER`, Predicted label: `O`
2. `Let's march together...`
   Given label: `LOC`, Predicted label: `O`
3. `Nastja Rysich (Germany) 3.75`
   Given label: `LOC`, Predicted label: `O`
4. `...from the Moslem, Arabised north.`
   Given label: `LOC`, Predicted label: `O`
5. `Mayor Antonio Gonzalez Garcia...`
   Given label: `PER`, Predicted label: `O`
6. `Spring Chg Hrw 12pct Chg White Chg`
   Given label: `LOC`, Predicted label: `O`
7. `...Prince Rainier told Reuters`
   Given label: `PER`, Predicted label: `O`
8. `Danila 28.5 16/12 Caribs/ up W224 Mobil.`
   Given label: `O`, Predicted label: `LOC`
9. `...next Wednesday.`
   Given label: `ORG`, Predicted label: `O`
10. `...Sale Limits US / UK / JP / FR`
    Given label: `LOC`, Predicted label: `O`

More than half of these sentences contain tokens that are incorrectly labeled. As shown above, some examples are ambiguous and may require more thoughtful handling. The dataset also contains edge cases such as punctuation characters like "/" that should be carefully handled.

F Label Error Detection on the Token-level

While our main focus was identifying sentences likely to contain a mislabeled token (given that reviewing an individual token's label anyway requires reading the broader context contained in the sentence), here we examine how accurately different estimates identify the erroneous tokens themselves. Recall from Section 2, we can use predicted class probabilities for each token from our trained model to compute a label quality score for each token q_i. We study three different label quality-scores (`sc`, `nm`, `cwe`) considered by Kuan and Mueller [10] to identify label errors in multi-class classification tasks.

After computing these scores for every token in the CoNLL-2003 test set (unmerged, ie. keeping the `B-` and `I-` tags), we evaluate token-level label error detection by sorting the tokens according to these scores (regardless of which sentence the tokens belonged to) and report the precision/recall for detecting which of these tokens were actually mislabeled or not (utilizing CoNLL++). Table 7 and Figure 2 contain the same precision/recall metrics described in Section 3, now computed at the token-level rather than sentence-level. `sc` and `nm` give the best performance (according to AUPRC or Lift metrics) for detecting which tokens are mislabeled. Unlike standard multi-class classification datasets where Kuan and Mueller [10] found `cwe` to be an effective label quality score to flag out-of-distribution examples for which no class label is appropriate, CoNLL-2003 and other entity recognition datasets contain an "other" class that explicitly accounts for tokens which do not belong to one of the classes of interest.

Table 7: Evaluating token-level label quality scores q_i on the CoNLL-2003 test set.

	AUPRC	AUROC	Lift @ # errors
`sc`	0.3483	0.9545	62.487
`nm`	0.3296	0.9552	66.362
`cwe`	0.2293	0.9355	46.986

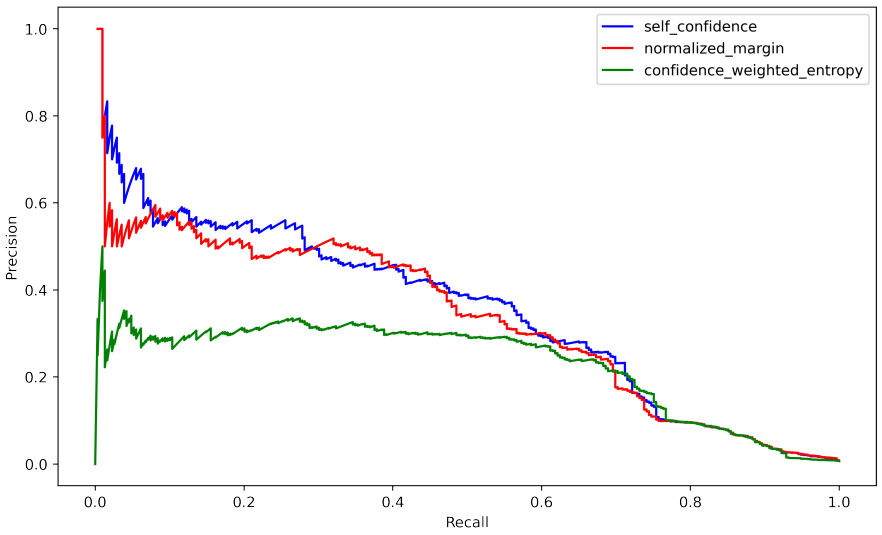


Figure 2: Precision-Recall curve for three different label quality scoring methods on the token level.