CrossWeigh: Training Named Entity Tagger from Imperfect Annotations

Zihan Wang[*] **Jingbo Shang**[*] **Liyuan Liu**[*] **Lihao Lu** **Jiacheng Liu** **Jiawei Han**

University of Illinois at Urbana-Champaign, Urbana, IL, USA

{zihanw2, shang7, ll2, lihaolu2, jl25, hanj}@illinois.edu

Abstract

Everyone makes mistakes. So do human annotators when curating labels for named entity recognition (NER). Such label mistakes might hurt model training and interfere model comparison. In this study, we dive deep into one of the widely-adopted NER benchmark datasets, CoNLL03 NER. We are able to identify label mistakes in about 5.38% test sentences, which is a significant ratio considering that the state-of-the-art test F_1 score is already around 93%. Therefore, we manually correct these label mistakes and form a cleaner test set. Our re-evaluation of popular models on this corrected test set leads to more accurate assessments, compared to those on the original test set. More importantly, we propose a simple yet effective framework, CrossWeigh, to handle label mistakes during NER model training. Specifically, it partitions the training data into several folds and train independent NER models to identify potential mistakes in each fold. Then it adjusts the weights of training data accordingly to train the final NER model. Extensive experiments demonstrate significant improvements of plugging various NER models into our proposed framework on three datasets. All implementations and corrected test set are available at our Github repo [1].

1 Introduction

Named entity recognition (NER), identifying both spans and types of named entities in text, is a fundamental task in the natural language processing pipeline. On one of the widely-adopted NER benchmarks, the CoNLL03 NER dataset (Sang and Meulder, 2003), the state-of-the-art NER performance has been pushed to a F_1 score around 93% (Akbik et al., 2019), through building end-to-end neural models (Lample et al., 2016; Ma and

---

[*] Equal Contributions.

[1] https://github.com/ZihanWangKi/CrossWeigh

Hovy, 2016) and introducing language models for contextualized representations (Peters et al., 2017, 2018; Akbik et al., 2018; Liu et al., 2018a). Such high performance makes the label mistakes in manually curated "gold standard" data non-negligible. For example, given a sentence "*Chicago won game 1 with Derrick Rose scoring 25 points.*", this "*Chicago*", representing the NBA team Chicago Bulls, should be annotated as an organization. However, when annotators are not careful or lack background knowledge, this "*Chicago*" might be annotated as a location, thus being a label mistake.

These label mistakes bring up two challenges to NER: (1) mistakes in the test set can interfere the evaluation results and even lead to an inaccurate assessment of model performance; and (2) mistakes in the training set can hurt NER model training. Therefore, in this paper, we conduct empirical studies to understand these mistakes, correct the mistakes in the test set to form a cleaner benchmark, and develop a novel framework to handle the mistakes in the training set.

We dive deep into the CoNLL03 NER dataset, and find label mistakes in about 5.38% test sentences. Considering that the state-of-the-art F_1 score on this test set is already around 93%, these 5.38% mistakes should be considered as significant. So we hire human experts to correct these label mistakes in the test set. We then re-evaluate recent state-of-the-art NER models on this new, cleaner test set. Compared to the results on the original test set, the re-evaluation results are more accurate and stable. Therefore, we believe this new test set can better reflect the performance of NER models.

We further propose a novel, general framework, CrossWeigh, to handle the label mistakes during the NER model training stage. Figure 1 presents an overview of our proposed framework. It contains two modules: (1) mistake estimation: it iden-
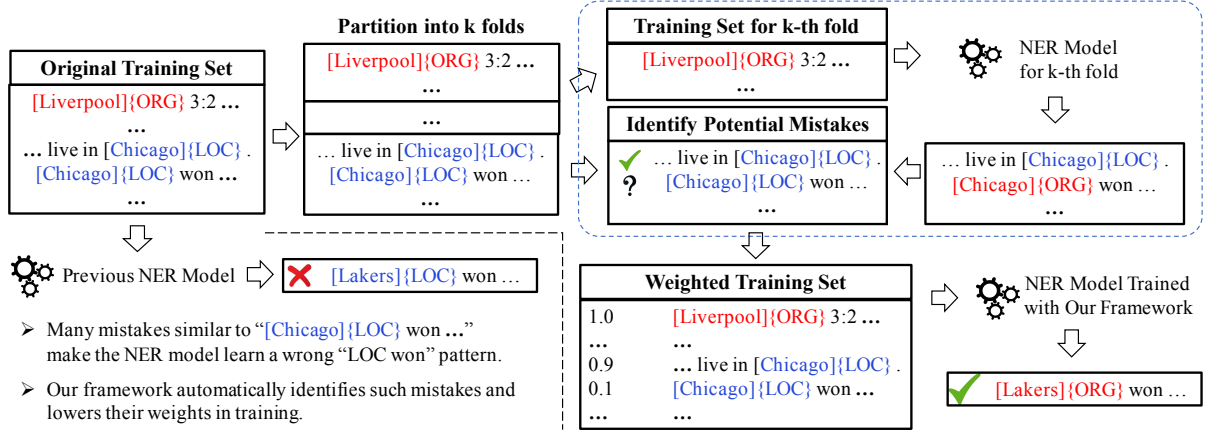
Figure 1: An overview of our proposed CrossWeigh framework. It can better handle label mistakes, identify low quality annotations and conduct learning from a weighted training set.

tifies the potential label mistakes in training data through a cross checking process and (2) mistake re-weighing: it lowers the weights of these instances during the training of the final NER model. The cross checking process is inspired by the k-fold cross validation; differently, in each fold's training data, it removes the data containing any of entities that appeared in this fold. In this way, each sentence will be scored by a NER model trained on a subset of training data not containing any entity in this sentence. Once we know where the potential mistakes are, we lower the weights of these sentences and train the final NER model based on this weighted training set. The final NER model is trained in a mistake-aware way, thus being more accurate. Note that, our proposed framework is general and fits most of, if not all, NER models that accept weighted training data.

To the best of our knowledge, we are the first to handle the label mistake issues systematically in the NER problem. We conduct extensive experiments on both the original CoNLL03 NER dataset and our corrected dataset. CrossWeigh is able to consistently improve performance when plugging with different NER models. In addition, we verify the effectiveness of CrossWeigh on emerging-entity and low-resource NER datasets. In summary, our major contributions are the following:

- We correct label mistakes in the test set of the CoNLL03 NER dataset and re-evaluate popular NER models. This establishes a more accurate NER benchmark.
- We propose a novel framework CrossWeigh to accommodate the mistakes during the model training stage. The proposed framework fits most

of, if not all, NER models.

- Extensive experiments demonstrate the significant, robust test F_1 score improvements of plugging NER models into our proposed framework on three datasets, not only CoNLL03 but also emerging-entity and low-resource datasets.

Reproducibility. We release both the corrected test set and the implementation of CrossWeigh framework[2].

2 CoNLL03 NER Re-Examination

The CoNLL03 NER dataset is one of the widely-adopted NER benchmark datasets. Its annotation guideline is based on MUC Conventions[3] (Sang and Meulder, 2003). Following this guideline, the annotators are asked to mark entities of person (PER), location (LOC), and organization (ORG), while using an extra miscellaneous (MISC) type to deal with entities that do not fall in these categories. This dataset has been split into training, development, and test sets, with 14041, 3250, and 3453 sentences, respectively.

2.1 Test Set Correction

In order to understand and correct the label mistakes, we have hired 5 human experts as annotators. Before looking at the data, we first train the annotators by carefully going through the aforementioned guideline. During the correction process, we strongly encourage the annotators to use search engines for suspicious token spans. This helps them

---

[2] https://github.com/ZihanWangKi/CrossWeigh

[3] https://www-nlpir.nist.gov/related_projects/muc/proceedings/ne_task.html

Sentence	Original labels	Corrected labels
Sporting Gijon 15 4 4 7 15 22 16	[Sporting]{ORG}	[Sporting Gijon]{ORG}
SOCCER - JAPAN GET LUCKY WIN , CHINA IN SURPRISE DEFEAT .	[JAPAN]{LOC}, [China]{PER}	[JAPAN]{LOC}, [China]{LOC}
NZ 's Bolger says Nats to meet NZ First on Sunday .	[NZ]{LOC}, [Bolger]{PER}, [Nats]{PER}, [NZ]{LOC}	[NZ]{LOC}, [Bolger]{PER}, [Nats]{ORG}, [NZ First]{ORG}
Seagramd ace 20/11/96 5,000 Japan	[Seagramd] {MISC}, [Japan]{LOC}	[Seagramd ace]{MISC}, [Japan] {LOC}

Table 1: Typical Examples of Our Corrections on the CoNLL03 NER dataset.

have more background knowledge. We also allow annotators to look at the original paragraph containing the sentence. This helps them have a better understanding of the context.

For the whole test set, we randomly split the test sentences between each pair combination of 5 annotators. In this way, each sentence in the test set is checked by exactly two annotators. The inter-annotator agreement is 95.66%. This is a reasonable score, given that the inter-annotator agreement in POS tagging annotations is about 97% (Manning, 2011). After we collected all annotations, we run a final round of verification on each sentence, where the original annotation and the two annotators' are not all the same. In the end, we have corrected label mistakes in 186 sentences, which is about 5.38% of the test set.

Table 1 presents some typical examples of our corrections. In the first sentence, as a sport team, "Sporting Gijon" was not annotated completely. In the second sentence, while "JAPAN" is correctly marked as LOC, "China" is wrongly identified as PER instead of LOC. One may notice that they both represent sport teams. However, according to the aforementioned guideline, country names should be marked as LOC even when they are sports teams. More details about this type of labels are discussed in Section 5. In the third sentence, "NZ" is the abbreviation of New Zealand. However, "Nat" and "NZ First" in fact refer to political parties (i.e., New Zealand Young Nationals and New Zealand First). So they should be labelled as ORG. In the forth sentence, looking at its paragraph, our annotators figure out that this is a table about ships and vessels loading items at different locations. Through comparing with other sentences in the context, such as "Algoa Day 21/11/96 6,000 Africa", our annotators identified "Seagramd ace" as a vessel, thus marking it as MISC. We have verified that there is indeed a vessel called "Seagrand Ace" ("Seagramd ace" might be a typo).

2.2 CoNLL03 Re-Evaluation

NER Algorithms. We re-evaluate following popu-

Method	Original	Corrected
LSTM-CRF	90.64 (±0.23)	91.47 (±0.15)
LSTM-CNNs-CRF	90.65 (±0.57)	91.87 (±0.50)
VanillaNER	91.44 (±0.16)	92.32 (±0.16)
Elmo	92.28 (±0.19)	93.42 (±0.15)
Flair	92.87 (±0.08)	93.89 (±0.06)
Pooled Flair	93.14 (±0.14)	94.13 (±0.11)

Table 2: CoNLL03 Re-Evaluation: Test F_1 scores and standard deviations on both original and corrected datasets. The results are based on 5 different runs.

lar NER algorithms:

- **LSTM-CRF** (Lample et al., 2016) incorporates long short term memory (LSTM) neural network with conditional random field (CRF). It also uses a word-wise character LSTM.
- **LSTM-CNNs-CRF** (Ma and Hovy, 2016) has a similar structure as LSTM-CRF, but captures character-level information through a convolutional neural network (CNN) over the character embedding.
- **VanillaNER** (Liu et al., 2018a) also extends LSTM-CRF and LSTM-CNNs-CRF by using a sentence-wise character LSTM.
- **ELMo** (Peters et al., 2018) extends LSTM-CRF and leverages pre-trained word-level language models for better contextualized representations.
- **Flair** (Akbik et al., 2018) also aims for contextualized representations, utilizing pretrained character level language models.
- **Pooled-Flair** (Akbik et al., 2018) extends Flair and maintains an embedding pool for each word to bring in dataset-level word embedding.

We use the implementation released by the authors for each algorithm and report the performance on original test set and corrected test set averaging 5 runs.

Results & Discussions. We re-evaluate the performance of the NER algorithms on the corrected test set. Their performance on the original test set is also listed for the reference. From the results in Table 2, one can observe that all models have higher F_1 scores as well as smaller standard deviations on the corrected test set, compared to those on the orig-

inal test set. Moreover, LSTM-CRF has a similar performance as LSTM-CNNs-CRF on the original test set, but on average lower performance on the corrected test set. This indicates that the corrected test set may be more discriminative. Therefore, we believe this corrected test set can better reflect the accuracy of NER algorithms in a stable way.

3 Our Framework: CrossWeigh

In this section, we introduce our framework. It is worth mentioning that our framework is designed to be general and fits most of, if not all, NER models. The only requirement is the capability to consume weighted training set.

3.1 Overview

As we have seen in the Section 2, human curated NER datasets are by no means perfect. Label mistakes in the training set can directly hurt the model's performance. As shown in Figure 1, if there are many similar mistakes like wrongly annotating "Chicago" in "Chicago won ..." as LOC instead of ORG, the NER model will likely capture the wrong pattern "LOC won" and make wrong predictions in future.

Our proposed CrossWeigh framework automates this process. Figure 1 presents an overview. It contains two modules: (1) mistake estimation: it identifies the potential label mistakes in training data through a cross checking process and (2) mistake re-weighing: it lowers the weights of these instances for the NER model training. The workflow is summarized in Algorithm 1.

3.2 Preliminary

We denote the training sentences as $\{x_1, x_2, \ldots, x_n\}$ where n is the number of sentences. Each sentence x_i is formed up of a sequence of words. Correspondingly, the label sequence for each sentence is denoted as $\{y_1, y_2, \ldots, y_n\}$. We use $\mathbf{D}$ to denote the training set, including both sentences and their labels. We use w_i to represent the weight of the i-th sentence. In most NER papers, the weights are uniform, i.e., $w_i = 1$.

We use $f(\mathbf{D}, w)$ to describe the training process of an NER model using the training set $\mathbf{D}$ weighted by w. This training process will return an NER model $M = f(\mathbf{D}, w)$. During this training, the

---

Algorithm 1: Our CrossWeigh Framework

Input: A NER model f, the training set $\mathbf{D} = \langle\{x_1, \ldots, x_n\}, \{y_1, \ldots, y_n\}\rangle$, and hyper-parameters k, t, and ϵ.
Output: A final NER model
for *i = 1 ... n* **do**
  | $c_i \leftarrow 0, w_i \leftarrow 1$
for *iter = 1 ... t* **do**
  | Randomly partition $\mathbf{D}$ into k folds.
  | **for** *Each fold D_i* **do**
  | | Obtain test_entities$_i$. (Eq. 2)
  | | Build train_set$_i$. (Eq. 3).
  | | Train a NER model
  | | $M_i = f(\text{train\_set}_i, w)$.
  | | **for** *Each $x_j \in D_i$* **do**
  | | | $\hat{y}_j \leftarrow M_i$'s prediction on x_j.
  | | | **if** $y_j \neq \hat{y}_j$ **then**
  | | | | $c_i \leftarrow c_i + 1$
for *i = 1 ... n* **do**
  | Compute w_i (Eq. 4).
Return $f(\mathbf{D}, w)$.

---

weighted loss function is as below.

$$\mathcal{J} = \sum_{i=1}^{n} w_i \cdot l(M(x_i), y_i) \qquad (1)$$

where $l(M(x_i), y_i)$ is the loss function of prediction $M(x_i)$ against its label sequence y_i. Typically, it is the negative log-likelihood of the model's prediction $M(x_i)$ compared to labeling sequence y_i.

3.3 Mistake Estimation

Our mistake estimation module is designed to let an NER model itself decide which sentences contain mistake and which do not. We would like to find sentences with label mistakes as many as possible (i.e. high recall), while keeping away from wrongly identified non-mistake sentences (i.e. high precision).

The basic idea of our mistake estimation module is similar to k-fold cross validation, however, in each fold's training data, it further removes the data containing any of entities appearing in this fold. The details are presented as follows.

We first randomly partition the training data into k folds: $D_1, D_2, \ldots, D_k$.

We then train k NER models separately based on these k folds. The i-th ($1 \leq i \leq k$) NER model M_i will be evaluated on the sentences in the hold-out fold D_i.

During its training, we avoid any sentence that may lead to "easy prediction" on this hold-out set.

	Original CoNLL03		Corrected CoNLL03	
	w/o CrossWeigh	w/ CrossWeigh	w/o CrossWeigh	w/ CrossWeigh
VanillaNER	91.44 ($\pm$0.16)	**91.78** ($\pm$**0.06**)	92.32 ($\pm$0.16)	**92.64** ($\pm$**0.08**)
Flair	92.87 ($\pm$**0.08**)	**93.19** ($\pm$0.09)	93.89 ($\pm$**0.06**)	**94.18** ($\pm$**0.06**)
Pooled-Flair	93.14 ($\pm$0.14)	**93.43** ($\pm$**0.06**)	94.13 ($\pm$0.11)	**94.28** ($\pm$**0.05**)

Table 3: Test F_1 scores and its standard deviations of models trained without or with CrossWeigh.

Therefore, we inspect every sentence in D_i and get the set of entities as follows.

$$\text{test\_entities}_i = \bigcup_{x_j \in D_i} e_j \qquad (2)$$

where e_j is the set of named entities in sentence x_j. We only consider the surface name in this entity set. That is, no matter "Chicago" is LOC or ORG, it only counts as its surface name "Chicago".

All training sentences that have entities included in test_entities$_i$ will be excluded in training process of the model M_i. Specifically,

$$\text{train\_set}_i = \{x_j | e_j \cap \text{test\_entities}_i = \emptyset, 1 \leq j \leq n\} \qquad (3)$$

We call this step as *entity disjoint filtering*. The intuition behind this step is that we want the model to make prediction of an entity without prior information of the entity itself from training. This will be helpful to detect sentences that are inconsistent.

We train k models M_i by feeding each train_set$_i$ into $f(\cdot, \cdot)$ with default uniform weight, and we use each M_i to make predictions for D_i and check for each sentence, whether the original label is the same as the model output. In this way, if the trained model M_i makes correct predictions on some sentences in D_i, they are more likely mistake-free. For those sentences that have labels disagreeing with the model output, we mark them as *potentially mistake*.

We run this mistake estimation module multiple iterations (i.e. t iterations) using different random partitions. Then, for each sentence in the training set, we get t estimations for it. We denote c_i ($0 \leq c_i \leq t$) as the confidence that sentence x_i contains label mistakes. c_i is defined as the the number of *potentially mistake* indications among all t estimations.

The number of folds k plays the role of a trade-off between the efficiency of the mistake estimation process and the number of training examples that can be used in each M_i. When k becomes larger, each fold D_i will be smaller, thus leading to a smaller size of test_entities$_i$; correspondingly, a larger train_set$_i$ will be picked. The model can

therefore be trained with more examples. However, it also slows down the whole mistake estimation process. On the CoNLL03 NER dataset, we observe that $k = 10$ leads to effective results, while having a reasonable running time.

3.4 Mistake Reweighing

In the mistake reweighing module, we adjust weight w_i for each sentence x_i that is marked as *potentially mistake* in the mistake estimation step. Here, we assign a weight w_i to all sentences marked, while the weights of other sentences remain 1. Specifically, we set $\forall i (1 \leq i \leq n)$,

$$w_i = \epsilon^{c_i} \qquad (4)$$

where ϵ is a parameter. In practice, it can be chosen according to the quality of mistake estimation module. Particularly, we first estimate the precision of the detected mistakes of a single iteration. Let p be the ratio of the number of true detected label mistakes over the number of detected label mistakes. p can be roughly estimated through a manual check of a random sample from the detected label mistakes. Then, we choose $\epsilon = 1 - p$, because $1 - p$ represents the fraction of these detected label mistakes that might be still useful during the model training. Therefore, for the sentences that are marked as *potentially mistake* in that iteration, ϵ of them are actually correct. With more iterations, the confidence of being correct lowers like a binomial distribution, which is the reason that we chose an exponential decaying weight function in Equation 4.

4 Experiments

In this section, we conduct several experiments to show effectiveness of our CrossWeigh framework. We first evaluate the overall performance of CrossWeigh on benchmark NER datasets, by plugging it into three base NER models. Since we have two modules in CrossWeigh, we then dive into each module and explore different variants and ablations. In addition, we further verify the effectiveness of CrossWeigh on two more datasets:

an emerging-entity NER dataset from WNUT'17 and a low-resource language NER dataset of the Sinhalese language.

4.1 Experimental Settings

Dataset. We use both the original and corrected CoNLL03 datasets. We follow the standard train/dev/test splits and use both the train set and dev set for training (Peters et al., 2017; Akbik et al., 2018). Entity-wise F_1 score on the test set is the evaluation metric.

Base NER Algorithm. We mainly choose Flair as our base NER algorithm. Flair is a strong NER algorithm using external resources (large corpus to train a language model). While Pooled-Flair has even better performance, its computational cost refrains us from doing extensive experiments.

Default Parameters in CrossWeigh. For all NER algorithms we experiment with, their default parameters are used. For CrossWeigh parameters, by default, we set $k = 10$, $t = 3$, and $\epsilon = 0.7$. We decide $\epsilon = 0.7$ because among 100 randomly sampled sentences with *potentially mistake*, we find that 27 of them really contain label mistakes (i.e., the probability of one annotation to be correct is roughly 70%). We use both train and development set to train the models, and report average F_1 and its standard deviation on both original test set and our corrected test set across 5 different runs (Peters et al., 2017).

4.2 Overall Performance

We pair CrossWeigh with our base algorithm (i.e. Flair) and two best-performing NER algorithms with or without language models in Table 2 (i.e. Pooled-Flair and VanillaNER), and evaluate their performance. As shown in Table 3, compared with the three algorithms, applying CrossWeigh always leads to a higher F_1 score and a comparable, sometimes even smaller, standard deviation. Therefore, it is clear that CrossWeigh can improve the performance of NER models. The smaller standard deviations also imply that the models trained with CrossWeigh are more stable. All these results illustrate the superiority of training with CrossWeigh.

4.3 Ablations and Variants

We pick Flair as the base algorithm to conduct ablation study.

Entity Disjoint Filtering. There is an entity disjoint filtering step, when we are collecting training

Method	Original	Corrected
w/o CrossWeigh	92.87 ($\pm$0.08)	93.89 ($\pm$0.06)
w/ CrossWeigh	93.19 ($\pm$0.09)	94.18 ($\pm$0.06)
− Entity Disjoint	92.88 ($\pm$0.11)	93.84 ($\pm$0.08)
+ Random Discard	93.01 ($\pm$0.10)	93.94 ($\pm$0.10)

Table 4: Importance of Entity Disjoint Filtering.

data train_set$_i$ for the NER model M_i during the mistake estimation step. To study its importance, we have done a few ablation experiments.

We have evaluated the following variants:

- Flair w/ CrossWeigh **– Entity Disjoint**: Skip the entity disjoint filtering step.
- Flair w/ CrossWeigh **+ Random Discard**: Instead of entity disjoint filtering, randomly discard the same number of sentences from each train_set$_i$ as it would do.

The results are listed in Table 4. One can easily observe that without the entity disjoint filtering, the F_1 scores are very close to the raw Flair model. This demonstrates that the entity disjoint filtering is critical to reduce the over-fitting risk in the mistake estimation step. Also, our proposed entity disjoint filtering strategy works more effective than random discard. This further confirms the effectiveness of entity disjoint filtering.

Variants in Computing c_i. There is definitely more than one way to determine c_i. Let δ be the number of "potentially mistake"'s among the t estimations, we can apply any of the following heuristics:

- **Ratio**: c_i is the number of "potentially mistake" (i.e. $c_i = \delta$). This is the method mentioned in Section 3, and used by default.
- **At Least One**: c_i is the indicator of at least one estimation being "potentially mistake" (i.e. $c_i = t \iff \delta >= 1$).
- **Majority**: c_i is the indicator of at least $\lfloor t/2 \rfloor + 1$ estimations being "potentially mistake" (i.e. $c_i = t \iff \delta >= \lfloor t/2 \rfloor + 1$).
- **All**: c_i is the indicator of all t estimations being "potentially mistake" (i.e. $c_i = t \iff \delta = t$).

We evaluate the performance of these heuristics when used in CrossWeigh, as shown in Table 5. There is not much difference across these heuristics, while our default choice "Ratio" is the most stable.

4.4 Label Mistake Identification Results

Another usage of CrossWeigh is to identify potential label mistakes during label annotation process, thus improving the annotation quality. This could

Heuristic	Original	Corrected
At Least One	93.10 (±0.10)	94.16 (±0.07)
Majority	93.20 (±0.09)	94.12 (±0.07)
All	93.16 (±0.09)	94.11 (±0.09)
Ratio	93.19 (±0.09)	94.18 (±0.06)

Table 5: Different c_i Estimation Heuristics.

t	Original	Corrected
1	93.09 (±0.14)	94.07 (±0.09)
5	93.23 (±0.10)	94.14 (±0.08)
3	93.19 (±0.09)	94.18 (±0.06)

Table 6: Different Numbers of Iterations t.

k	Original	Corrected
2	92.11 (±0.24)	92.88 (±0.11)
5	93.12 (±0.08)	94.12 (±0.08)
10	93.19 (±0.09)	94.18 (±0.06)

Table 7: Different Numbers of Folds k.

ϵ	Original	Corrected
0.3	92.79 (±0.14)	93.68 (±0.15)
0.5	93.21 (±0.09)	94.18 (±0.07)
0.9	93.01 (±0.10)	93.96 (±0.09)
0.7	93.19 (±0.09)	94.18 (±0.06)

Table 8: Different Weight Adjustments ϵ.

be also helpful to active learning.

Specifically in this experiment, we apply our noise estimation module to the concatenation of training and testing data. As we have manually corrected the label mistakes in the testing set, we are able to report the number of true mistakes among the potential mistakes discovered in the test set.

The results are presented in Table 9. The potential mistakes are the total number of mistakes identified by CrossWeigh, and actual mistakes is the true positives among all identifications. From the results, we can see that when the base model is Flair, CrossWeigh is able to spot more than 75% of label mistakes, while maintaining a precision about 25%. It is worth noting that 25% is a reasonably high precision, given that the label mistake ratio is only 5.38%. The 75% recall indicates that CrossWeigh is able to identify most of the label mistakes, which are extremely valuable to improve the annotation quality.

4.5 Parameter Study

We study how CrossWeigh performs with different hyper-parameters, i.e., t (the number of iterations that we run mistake estimation), k (the number of folds in mistake estimation), and ϵ (the weight scaling factor of identified potential mistakes).

In principle, a larger t usually gives us a more stable mistake estimation. However, a larger t also requires more computation resources. In our experiments (see Table 6), we find that $t = 3$ provides a good enough result.

Specifically, during mistake estimation, we have to choose the number of folds to partition the data. The more partitions made, the smaller each D_i is and the fewer sentences will be filtered, leading to more training data train_set$_i$ and better trained

M_i. On the other hand, this is at the cost of higher computational expense. As shown in Table 7, we observe that $k = 5, k = 10$ are significantly better than $k = 2$. In fact, when $k = 2$, each train_set$_i$ has only around 5000 sentences and 1500 entities inside. These numbers become 7000 and 4000 when $k = 5$, and 9000 and 7000 when $k = 10$.

As we mentioned before, the value ϵ can be chosen by estimating the quality of mistake estimation. Table 8 presents some results when other values are used. $\epsilon = 0.3$ leads to the worst performance. Since our estimation does not have high precision, assigning ϵ to a low value like 0.3 may not be a good choice. Interestingly $\epsilon = 0.5$ performs on par with $\epsilon = 0.7$, and even slightly better in the original test set. We hypothesize that this is because there are some ambiguous sentences that we did not count during estimating the quality of mistake estimation, see Section 5, and the actual precision could be higher.

4.6 Other Datasets

To show the generalizability of our method across domains and languages, we further evaluate CrossWeigh on an emerging-entity NER dataset from WNUT'17 and a Sinhalese NER dataset from LORELEI[4]. Sinhalese is a low-resource, morphology-rich language. For WNUT'17, we use the Flair as our base NER algorithm. For Sinhalese, we use BERT (Devlin et al., 2018) followed by a BiLSTM-CRF as our base NER algorithm. We use the same parameters as used in the previous CoNLL03 experiments, namely $k = 10, t = 3, \epsilon = 0.7$.

The results averaged across 5 runs are reported in Table 10. One can observe quite similar results as those in the previous CoNLL03 experiments.

---
[4]LDC2018E57

Algorithm	Potential Mistakes	Actual Mistakes	Precision	Recall	F1
Flair	573.0	144.0	0.2513	0.7742	0.3794
VanillaNER	821.67	146.33	0.1781	0.7867	0.2904

Table 9: Quality of noise estimation. The number of true mistakes, based on our manual correction, is 186. The potential mistakes are counted based on average of 3 runs.

Dataset	w/o CrossWeigh	w/ CrossWeigh
WNUT'17	48.96 ($\pm$0.97)	50.03 ($\pm$0.40)
Sinhalese	66.34 ($\pm$0.34)	67.68 ($\pm$0.21)

Table 10: Applying CrossWeigh on other datasets

Training with CrossWeigh leads to a significantly higher F1 and a smaller standard deviation. This suggests that CrossWeigh works well in other datasets and languages.

5   Case Studies

Test Set Correction. Despite the label mistakes that we have corrected, we also find some ambiguous but consistent cases. For instances, (1) All NBA/NHL divisions such as "CENTRAL DIVISION", "WESTERN DIVISION" were annotated as MISC, while all European leagues, such as "SPANISH FIRST DIVISION" and "ENGLISH PREMIER LEAGUE", are not marked as MISC correctly — only "SPANISH" and "ENGLISH" are labelled as MISC. And (2) "Team A at Team B" is a way to say "Team A" as an away team playing with Team B as a home team. However, in almost all cases (only 1 exception out of more than 100), "Team A" was labelled as ORG while "Team B" was labelled as LOC. For example, in "MINNESOTA AT MILWAUKEE", "NEW YORK AT CALIFORNIA", and "ORLANDO AT LA LAKERS", the second sports team "MILWAUKEE", "CALIFORNIA" and "LA LAKERS" were always labelled as LOC. Because these parts behave consistently and generally follow the annotation guideline, we didn't touch them during the test set correction.

CrossWeigh Framework. The mistakes in the training set can harm the generalizability of the trained model. For example, in Table 11, the original training sentence "Hapoel Haifa 3 Maccabi Tel Aviv 1" contains a label mistake, because "Maccabi Tel Aviv" is a sports team but was not annotated completely. Interestingly, there is a similar sentence in the test set – "Hapoel Jerusalem 0 Maccabi Tel Aviv 4". In all 5 different runs of the original Flair model, they failed to predict correctly that

"Maccabi Tel Aviv" in the test sentence as ORG because of the label mistake in the training sentence, even though "ORG number ORG number" is an obvious pattern in the training set. In CrossWeigh, this label mistake in the training set was detected in all $t = 3$ iterations and therefore assigned a very low weight during training. After that, in all 5 different runs of Flair w/ CrossWeigh, they successfully predict that "Maccabi Tel Aviv" is ORG as a whole.

6   Related Work

In this section, we review related works from three aspects, mistake identification, cross validation & boosting, and NER algorithms.

6.1   Mistake Identification

Researchers have noticed the label mistakes in sophisticated natural language processing tasks for a while. For example, it is reported that the inter-annotator agreement is about 97% on the Penn Treebank POS tagging dataset (Manning, 2011; Subramanya et al., 2010).

There are a few attempts towards detecting label mistakes automatically. For example, Nakagawa and Matsumoto (2002) designed a support vector machine-based model to assign weights to examples that were hard to classify in the POS tagging task. Helgadóttir et al. (2014) further applied previous detection models and manually corrected Icelandic Frequency Dictionary (Pind et al., 1991) POS tagging dataset. However, these two methods are specifically developed for POS tagging and cannot be directly applied to NER.

Recently, Rehbein and Ruppenhofer (2017) extends variational inference with active learning to detect label mistakes in "silver standard" data generated by machines. In this paper, we focus on detecting label mistakes in "gold standard" data, which is a different scenario.

6.2   Cross Validation & Boosting

Our mistake estimation module shares some similarity with cross validation. Applying cross valida-

	Training Set	Test Set
Text	Hapoel Haifa 3 Maccabi Tel Aviv 1	Hapoel Jerusalem 0 Maccabi Tel Aviv 4
Original Annotations	[Hapoel Haifa]{ORG}, **[Tel Aviv]{ORG}**	[Hapoel Jerusalem]{ORD}, [Maccabi Tel Aviv]{ORG}
Correct Annotations	[Hapoel Haifa]{ORG}, [Maccabi Tel Aviv]{ORG}	[Hapoel Jerusalem]{ORD}, [Maccabi Tel Aviv]{ORG}

	Action	Result
Flair	Assumes this sentence is equally reliable as others.	[Hapoel Jerusalem]{ORD}, **[Tel Aviv]{ORG}**
Flair w/ CrossWeight	Lowers the weight of this sentence as mistakes.	[Hapoel Jerusalem]{ORD}, [Maccabi Tel Aviv]{ORG}

Table 11: Case Study on the CoNLL03 dataset. Errors are marked with red

tion to the training set is the same as our mistake estimation module, except that we have an *entity disjoint filtering* step. Experiments in Table 4 show that this step is crucial to our performance gain. The choice of ten folds also stems from cross validation (Kohavi, 1995).

Another similar thread of work is boosting, such as Adaboost (Freund et al., 1999; Schapire and Singer, 1999). For example, Abney et al. (1999) has applied Adaboost on the Penn Treebank POS tagging dataset and gained encouraging results on model performance. In boosting algorithms, the training data is assumed to be perfect. Therefore, it trains models using the full training set and then increases the weights of training instances that fails the current model in the next round of learning. In contrast, we decrease the weights of sentences that differ from the model built upon the entity disjoint training set. More importantly, our framework is a better fit for neural models, because they can likely overfit the training data and thus being bad choices as weak classifiers in boosting.

6.3 NER Algorithms

Neural models have been widely used for Named Entity Recognition, and the state-of-the-art models integrate LSTMs, conditional random field and language models (Lample et al., 2016; Ma and Hovy, 2016; Liu et al., 2018b; Peters et al., 2018; Akbik et al., 2018). In this paper, we focus on improving the annotation quality for NER, and our method has a big potential to help other methods, especially for noisy datasets (Shang et al., 2018).

7   Conclusion & Future work

In this paper, we explore and correct the label mistakes in the CoNLL03 NER dataset. Based on the corrected test set, we re-evaluate most of recent NER models. We further propose a novel framework, CrossWeigh, that is able to detect label mistakes in the training set and then train a more robust NER model accordingly. Extensive experiments demonstrate the effectiveness of CrossWeigh on three datasets and also indicate the potentials of using CrossWeigh to improve the annotation quality during the label curation process.

In future, we plan to extend our framework into an iterative setting, similar to those boosting algorithms. The bottleneck of doing this lies in the efficiency problems of training multiple deep neural models hundreds of times. One solution to overcome it is to apply meta learning. We can first train a meta model and only fine-tune on different training data on each fold. In this way, we can identify label mistakes more accurately and obtain a series of weighted models at the end.

References

Steven Abney, Robert E Schapire, and Yoram Singer. 1999. Boosting applied to tagging and pp attachment. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition. In *NAACL*.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1638–1649.

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

Yoav Freund, Robert Schapire, and Naoki Abe. 1999. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612.

Sigrún Helgadóttir, Hrafn Loftsson, and Eiríkur Rögnvaldsson. 2014. Correcting errors in a new gold standard for tagging icelandic text. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014.*, pages 2944–2948.

Ron Kohavi. 1995. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, pages 1137–1145. Morgan Kaufmann.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016*, pages 260–270.

Liyuan Liu, Xiang Ren, Jingbo Shang, Xiaotao Gu, Jian Peng, and Jiawei Han. 2018a. Efficient contextualized representation: Language model pruning for sequence labeling. In *EMNLP*, pages 1215–1225. Association for Computational Linguistics.

Liyuan Liu, Jingbo Shang, Xiang Ren, Frank Fangzheng Xu, Huan Gui, Jian Peng, and Jiawei Han. 2018b. Empower sequence labeling with task-aware neural language model. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Xuezhe Ma and Eduard H. Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Christopher D. Manning. 2011. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In *Computational Linguistics and Intelligent Text Processing - 12th International Conference, CICLing 2011, Tokyo, Japan, February 20-26, 2011. Proceedings, Part I*, pages 171–189.

Tetsuji Nakagawa and Yuji Matsumoto. 2002. Detecting errors in corpora using support vector machines. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL (1)*, pages 1756–1765. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*, pages 2227–2237. Association for Computational Linguistics.

Jörgen Pind, Friðrik Magnússon, and Stefán Briem. 1991. Íslensk orðtíðnibók [the icelandic frequency dictionary]. *The Institute of Lexicography, University of Iceland, Reykjavik, Iceland*.

Ines Rehbein and Josef Ruppenhofer. 2017. Detecting annotation noise in automatically labelled data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1160–1170.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 142–147.

Robert E Schapire and Yoram Singer. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336.

Jingbo Shang, Liyuan Liu, Xiaotao Gu, Xiang Ren, Teng Ren, and Jiawei Han. 2018. Learning named entity tagger using domain-specific dictionary. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2054–2064.

Amarnag Subramanya, Slav Petrov, and Fernando C. N. Pereira. 2010. Efficient graph-based semi-supervised learning of structured tagging models. In *EMNLP*, pages 167–176. ACL.