# Training Named Entity Tagger From Imperfect Annotations

Ruoyuan Du
Department of Computer Science
Tufts University
Medford, MA 02155
`ruoyuan.du@tufts.edu`
Zeyu Chang
Department of Computer Science
Tufts University
Medford, MA 02155
`zeyu.chang@tufts.edu`
Yuwei He
Department of Computer Science
Tufts University
Medford, MA 02155
`yuwei.he@tufts.edu`

May 15, 2021

**Abstract**

With the growing use of named entity recognition (NER), label mistakes happen during the annotation. Such mistakes can hinder model performance and interfere model training. Recent studies show that such labels require more training steps and are more frequently forgotten than clean labels, thus they're also easy to identify in training. Inspired by the CrossWeigh framework(Wang et al., 2019c), we combine the algorithm with the flair model to improve the accuracy that's hurt by the incorrect labels. Extensive experiments on two widely used but noisy benchmarks for information extraction, SEC and CoNLL03, demonstrate the effectiveness of our framework.

## 1 Introduction

Named-entity recognition(NER), a sub-task of information extraction that seeks to locate and classify named entities mentioned in unstructured text into predefined categories, can help identify important information in the dataset, which is powerful when dealing with the huge dataset, and as a fundamental task in the natural language processing pipeline. On one of the widely used NER benchmarks, the CoNLL03 NER dataset [8], the high performance of the NER model makes the label mistakes influence the accuracy heavily. For example, given the 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[9].

During the annotation, a notable portion of incorrect labels has been caused, the incorrect labels can lead to bad performance. In specific, 1). The mistakes in the test set can lead to the inaccurate result of the evaluation, 2). The mistakes in the training set can hinder model training.

We take a closer look into the CoNLL03 NER dataset and find out a typical public data set has around 5% such labeling mistakes. With the research efforts from the CrossWeigh framework (Wang et al., 2019c), a representative work that improves a natural language model without using extra learning resources. This method trains multiple independent models on different partitions of training data, and manipulates the weighs instances.

In this paper, we aim to develop a framework based on the CrossWeigh that can improve the accuracy and corporate with the NER model. Our work is motivated by the flair deep neural network, which has a powerful text embedding library, it can combine different word and document embeddings. In specific, it included contextual string embeddings, character-level embedding, and stacked embeddings. Meanwhile, it also includes strong sequence labeling models, the CRF(Conditional Random Field), which can provide conditional probability over label sequences given a specific sequence of observations. In this way, we can incorporate the CrossWeigh algorithm with the flair model. The results demonstrate the effectiveness of our method in enhancing label accuracy training, we improved the f1 score, and pushed the score around 95%. In summary, our major contributions are the following:

• We propose a general co-regularization framework that can effectively learn supervised NER models from noisy datasets.

• We discuss in detail the different design strategies with f1 score as a training sign of the framework and the trade-off between efficiency and effectiveness.

• Extensive experiments on NER and Flair demonstrate that our framework results in promising improvements on models, and also outperforms the CrossWeigh frameworks.

## 2 Related Works

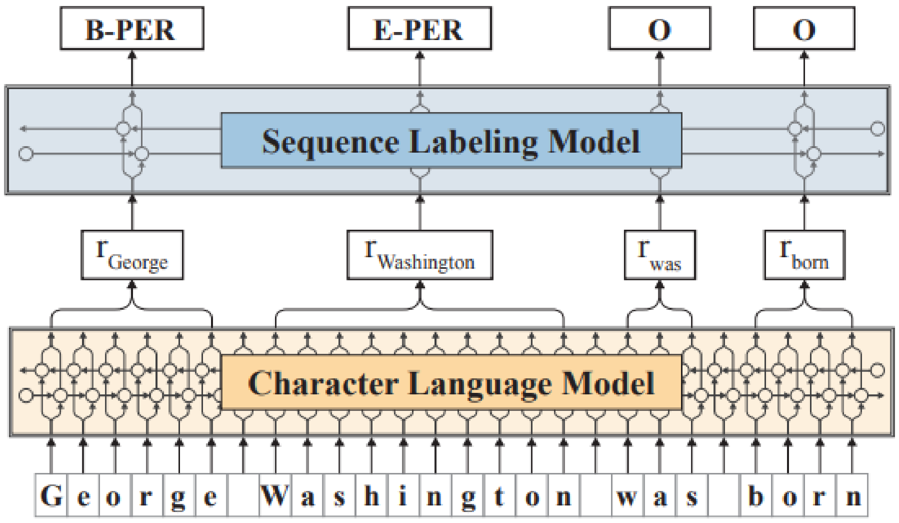
Labeling mistakes in data is one of the most harmful phenomena for machine learning model training since even the ground truth is wrong. This is also an issue that we cannot avoid since humans make mistakes, and to fix a data set with more than 10 million rows is both time-consuming and expansive. As machine learning becomes more and more popular, researchers and industrial experts found that fixing the mislabeling data is important and is a worthy field to research. There are different techniques for different machine learning models. For example, there is a framework to train Convolutions Neural Network to solve the noisy labeled image classification[10]. Besides, there is also a label noise model for text classification[4]. However, all above-mentioned model requires huge computational power and complex algorithm, which are not suitable for a Named Entity project. Hence, we introduce a relatively light-weighted framework, CrossWeigh[9], to deal with the imperfect annotation. We also modified the structure and run experiment to find out the most effective element it has

Developing a powerful Named Entity Recognition model is also the main goal for us in this project. With the deep learning technique, we can let the machine interpret sentences. There are many powerful Natural Lagrange Process models to classify tags for words and the word vectorization methods. For example, BERT [2] is one of the most well-known language models for both word embedding and learning. Glove[5] is also a popular and powerful word embedding pre-trained matrix based on the probability calculation. We will use Glove for our latter model training. In our work, we used Flair[1] as a based model and do the hyperparameter tuning and performance comparison.

## 3 Methodology

### 3.1 Model Explanation

We mainly used the flair NLP model[1] to make predictions. In this section, we will describe the flair structure and the algorithm we used for training a valid NER model with high accuracy.



This is the structure of the flair model, the bottom is the word-embedding step and the above is the sequence laeling model. Here, we used a LSTM-CRF to predict the tags.[1]

**Text Embedding** We can explain the Flair Text embedding in three aspects: 1. Contextual String embedding, 2. Character-level language modeling, 3. Stacked embedding.

Firstly, flair uses pre-trained unlabeled corpora and captures the meaning of a word according to its surrounding text. This is called Contextualized word embeddings intro ducted by [7]. Language models compute the probability distribution of the next word in a sequence given the sequence of previous words. Thus, as a model with long-term memory, LSTM is a perfect tool to train such a model. However, predict the words by just the previous token is not ideal. Therefore, researchers introduced the multi-layer LSTM [6] to let the model read the sentence both forward and backward. Thus, we have a text embedding that can understand the context.

Secondly, to support the Contextual String embedding, flair uses a character-level embedding. Instead of tokenizing each word, we tokenize each character. This technique is important since by doing the character-level vectorization, even the same word can have different number vectorization. Such property can significantly improve the accuracy of NER since many words can actually have different meanings. As the example stated in the introduction, Chicago can represent a City but it can also represent an Organization.

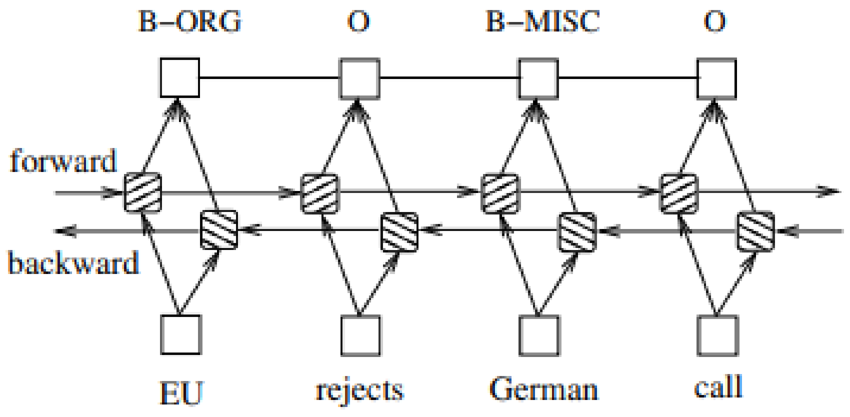
Thirdly, stacked embeddings are one of the most important concepts of flair. We can combine different embeddings together before training by simply concatenate them together. We can also see this is a technique to increasing the training set since we feed different embedding into the dnn model.

**LSTM-CRF** Recent studies show that LSTM-CRF can have high accuracy for sequence tagging[3].

Conditional Random Field(CRF) is a special case of Markov Random field. In the Hidden Markov field, the consequence is calculated by the joint probability of hidden status and the observed data. But in CRF, the conditional probability of the Hidden layer given by the observed data is calculated. Therefore, unlike the HMM, it is a discriminative model since the word is always given or observed and we only need to predict the tag, in this case, hidden status. The conditional probability is calculated by the following:

$$p(l|s) = \frac{exp[score(l|s)]}{\sum_{l'} exp[score(l'|s)]} = \frac{exp[\sum_{j=1}^{m}\sum_{i=1}^{n}\lambda_i f_j(s,i,l_i,l_{i-1})]}{\sum_{l'} exp[\sum_{i=1}^{n}\lambda_i f_j(s,i,l'_i,l'_{i-1})]} \quad (1)$$

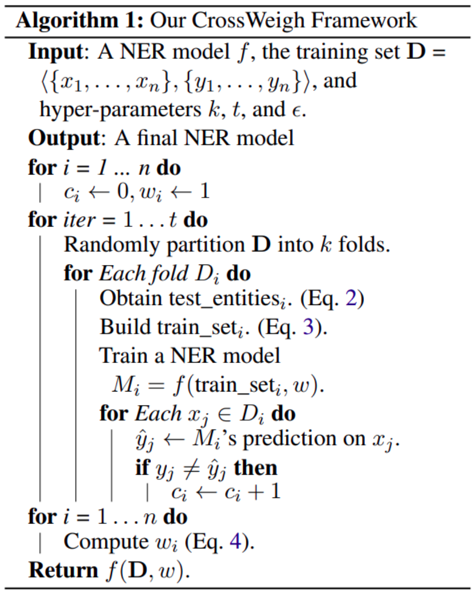
Then, we can combine a LSTM network and a CRF network to form a LSTM-CRF model.[3]. With the CRF layer, we can utilize the past and future tags to predict the current tag. Then, with LSTM network, we can do the same thing to predict the tags.



This figur shows the structure of LSTM-CRF model [3]. The shaded layer is bidirectional LSTM and the top layer is CRF network.

## 3.2 CrossWeigh

Label mistakes in the training set can directly hurt the model's performance. For example, if there are many similar mistakes like wrongly annotating "Chicago" in "Chicago won..." as LOC instead of ORG, then the NER will likely capture the wrong pattern "LOC won" and make wrong predictions.The proposed CrossWeigh framework can automatically identify potential mistakes by assigning different weights of these instances in NER model training, so that



```
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 $\epsilon$.
Output: A final NER model
for i = 1 ... n do
    $c_i \leftarrow 0, w_i \leftarrow 1$
for iter = $1 \ldots t$ 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
          $M_i = f$(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 = $1 \ldots n$ do
    Compute $w_i$ (Eq. 4).
Return $f(\mathbf{D}, w)$.
```

Figure 1: CrossWeigh Algorithm

these instances will contribute less to the loss function. The algorithm from the paper is presented in figure 1.

The training sentences are denoted as D, including both sentences and their labels. $w_i$ is denoted as the weight of the $i$-th sentence. $f(D, w)$ describe the training process of an NER model using the training set D weighted by w. The basic idea is to split the training sets into k folds and train k NER models and compare the predicted labels with actual labels. If the predicted does not equal to its actual label, the sentence will be marked as having potential mistake. This process will be iterated for t times. $c_i$ is the number of wrong predicted labels in t iterations $(1 \leq c_i \leq t)$. $w_i$ is calculated as $w_i = \epsilon^{c_i}$. Finally, the NER model will be re-trained with the weighted training set to be the final trained NER model.

With the algorithm, we modified this algorithm to exclude the weighing part and disjoint filtering part separately, to examine the efficiency of the two modules.

| Method | F1 score |
|---|---|
| w/o CrossWeigh | 94.83 |
| w/ CrossWeigh | 95.32 |
| - No weights | 95.24 |
| - No disjoint filtering | 94.82 |

Table 1: F1 score of using original CrossWeigh, modified CrossWeigh and without CrossWeigh

## 4 Experimental Setup & Result

### 4.1 Preprocessing: Disjoint filtering

To avoid easy prediction, first it is needed to collect all entities in test set as follows:

$$test\_entities_i = \cup_{x_j \in D_i} e_j$$

where $e_j$ is the set of named entities in sentence $x_j$. Then, all training sentences that have entities included in test_entities_i will be excluded in training process of the model $M_i$.

### 4.2 Split training data

Training set $D$ is splitted in to k folds. Then in each fold, we randomly split the data to have train and held-out data. The split ratios of held-out set is 0.2.

### 4.3 Compare the F1 score of NER model w/ and w/o CrossWeigh

We explored the efficacy of the modified framework and compare with the model with complete CrossWeigh framework and without the framework.

First we modify the algorithm to exclude the disjoint filtering part. Then the algorithm was modified to exclude the weighing part. All models are fed with the SEC ner dataset. F1 score is used as the evaluation metrics.

From the table 1 we can see that the NER model with CrossWeigh algorithm yields the best F1 score of 95.32. With the CrossWeigh algorithm but not the weights generates a F1 score of 96.24, which is close to the F1 score of using the full CrossWeigh algorithm. The F1 score of not using CrossWeigh is similar to the F1 score of using CrossWeigh without disjoint filtering, of 94.83 and 94.82 respectively. From this experiment we can see that the disjoint filtering step in the CrossWeigh plays very important part in improving the model's accuracy.

| epsilon | 0.1 | 0.3 | 0.5 | 0.7 | 0.9 |
|---|---|---|---|---|---|
| w/o CrossWeigh | 94.83 | 94.93 | 95.39 | 95.32 | 95.11 |

Table 2: F1 score of different epsilon value

### 4.4 Performance comparison of different $\epsilon value$

As $\epsilon$ being $\epsilon = 1 - p$, p is the ratio of the number of true detected label mistakes over the number of detected label mistakes. In the paper, the authors chose $\epsilon$ by manually checking the true detected label mistakes from the detected label mistakes, which is not practical and accurate when we have large quantities of data. So we tuned the $\epsilon$ values as a hyperparameter.The result is shown in table 2.
$\epsilon = 0.5$ yields the best F1 score in this setting, which is different from the $\epsilon$ value chosen in the paper. Therefore, when we have different dataset and models, $\epsilon$ should be treated as a hyperparameter instead of a fixed value.

## 5 Conclusion

In this paper, they explored and corrected the label mistakes in the CoNLL-03 NER dataset. Based on the corrected test set, they re-evaluated some basic NER models. Furthermore, they proposed the CrossWeigh framework which helped identify potential mistakes in training data and then trained a more robust NER model to improve prediction accuracy. Some experiments had shown the effectiveness of CrossWeigh, however, I doubted the effectiveness of assigning weights to sentences in this framework by checking some presented results. I would try to remove the weighting part in the framework and check the effectiveness.

## 6 References

## References

[1] Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. Flair: An easy-to-use framework for state-of-the-art nlp. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59, 2019.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[3] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991, 2015.

[4] Ishan Jindal, Daniel Pressel, Brian Lester, and Matthew S. Nokleby. An effective label noise model for DNN text classification. *CoRR*, abs/1903.07507, 2019.

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

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

[7] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018.

[8] Erik F. Tjong Kim Sang and Fien 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*, pages 142–147, 2003.

[9] Zihan Wang, Jingbo Shang, Liyuan Liu, Lihao Lu, Jiacheng Liu, and Jiawei Han. Crossweigh: Training named entity tagger from imperfect annotations. *arXiv preprint arXiv:1909.01441*, 2019.

[10] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.