Named Entity Recognition with Partially Annotated Training Data

Stephen Mayhew[♮], **Snigdha Chaturvedi**[♯], **Chen-Tse Tsai**[♭], **Dan Roth**[♮]
[♮]University of Pennsylvania, Philadelphia, PA, 19104
[♯]University of North Carolina, Chapel Hill, NC, 27599
[♭]Bloomberg LP
{mayhew, danroth}@seas.upenn.edu,
snigdha@cs.unc.edu, ctsai54@bloomberg.net

Abstract

Supervised machine learning assumes the availability of fully-labeled data, but in many cases, such as low-resource languages, the only data available is partially annotated. We study the problem of Named Entity Recognition (NER) with partially annotated training data in which a fraction of the named entities are labeled, and all other tokens, entities or otherwise, are labeled as non-entity by default. In order to train on this noisy dataset, we need to distinguish between the true and false negatives. To this end, we introduce a constraint-driven iterative algorithm that learns to detect false negatives in the noisy set and downweigh them, resulting in a weighted training set. With this set, we train a weighted NER model. We evaluate our algorithm with weighted variants of neural and non-neural NER models on data in 8 languages from several language and script families, showing strong ability to learn from partial data. Finally, to show real-world efficacy, we evaluate on a Bengali NER corpus annotated by non-speakers, outperforming the prior state-of-the-art by over 5 points F1.

1 Introduction

Most modern approaches to NLP tasks rely on supervised learning algorithms to learn and generalize from labeled training data. While this has proven successful in high-resource scenarios, this is not realistic in many cases, such as low-resource languages, as the required amount of training data just doesn't exist. However, partial annotations are often easy to gather.

We study the problem of using partial annotations to train a Named Entity Recognition (NER) system. In this setting, all (or most) identified entities are correct, but not all entities have been identified, and crucially, there are no reliable examples of the negative class. The sentence shown in Figure 1 shows examples of both a gold and a partially annotated sentence. Such partially annotated data is relatively easy to obtain: for example, a human annotator who does not speak the target language may recognize common entities, but not uncommon ones. With no reliable examples of the negative class, the problem becomes one of estimating which unlabeled instances are true negatives and which are false negatives.

			False Negative					
	Arsenal	hires	Unai	Emery	as	Arsene	Wenger's	successor
Gold	ORG	O	PER	PER	O	PER	PER	O
Partial	ORG	O	O	O	O	PER	PER	O
Weights								
Oracle	1	1	0	0	1	1	1	1
Our model	1	0.6	0.01	0.03	0.8	1	1	0.4

Figure 1: This example has three entities: *Arsenal*, *Unai Emery*, and *Arsene Wenger*. In the *Partial* row, the situation addressed in this paper, only the first and last are tagged, and all other tokens are assumed to be non-entities, making *Unai Emery* a false negative as compared to *Gold*. Our model is an iteratively learned binary classifier used to assign weights to each token indicating its chances of being correctly labeled. The *Oracle* row shows optimal weights.

To address the above-mentioned challenge, we present Constrained Binary Learning (CBL) – a novel self-training based algorithm that focuses on iteratively identifying true negatives for the NER task while improving its learning. Towards this end, CBL uses constraints that incorporate background knowledge required for the entity recognition task.

We evaluate the proposed methods in 8 languages, showing a significant ability to learn from partial data. We additionally experiment with initializing CBL with domain-specific instance-weighting schemes, showing mixed results. In the process, we use weighted variants of popular NER models, showing strong performance in both non-neural and neural settings. Finally, we show experiments in a real-world setting, by employing non-speakers to manually annotate romanized Bengali text. We show that a small amount of non-speaker annotation combined with our method can outperform previous methods.

2 Related Work

The supervision paradigm in this paper, partial supervision, falls broadly under the category of semi-supervision (Chapelle et al., 2009), and is closely related to weak supervision (Hernández-González et al., 2016)[1] and incidental supervision (Roth, 2017), in the sense that data is constructed through some noisy process. However, all of the most related work shares a key difference from ours: reliance on a small amount of fully annotated data in addition to the noisy data.

Fernandes and Brefeld (2011) introduces a transductive version of structured perceptron for partially annotated sequences. However, their definition of partial annotation is labels removed at random, so examples from all classes are still available if not contiguous.

Fidelity Weighted Learning (Dehghani et al., 2017) uses a teacher/student model, in which the teacher has access to (a small amount) of high quality data, and uses this to guide the student, which has access to (a large amount) of weak data.

Hedderich and Klakow (2018), following Goldberger and Ben-Reuven (2017), add a noise adaptation layer on top of an LSTM, which learns how to correct noisy labels, given a small amount of training data. We compare against this model in our experiments.

In the world of weak supervision, Snorkel (Ratner et al., 2017; Fries et al., 2017), is a system that combines automatic labeling functions with data integration and noise reduction methods to rapidly build large datasets. They rely on high recall and consequent redundancy of the labeling functions. We argue that in certain realistic cases, high-recall candidate identification is unavailable.

We draw inspiration from the Positive-Unlabeled (PU) learning framework (Liu et al., 2002, 2003; Lee and Liu, 2003; Elkan and Noto, 2008). Originally introduced for document classification, PU learning addresses problems where examples of a single class (for example, sports) are easy to obtain, but a full labeling of all other classes is prohibitively expensive.

Named entity classification as an instance of PU learning was introduced in Grave (2014), which uses constrained optimization with constraints similar to ours. However, they only address the problem of named entity classification, in which mentions are given, and the goal is to assign a *type* to a named-entity (like 'location', 'person', etc.) as opposed to our goal of identifying and typing named entities.

Although the task is slightly different, there has been work on building 'silver standard' data from Wikipedia (Nothman et al., 2008, 2013; Pan et al., 2017), using hyperlink annotations as the seed set and propagating throughout the document.

Partial annotation in various forms has also been studied in the contexts of POS-tagging (Mori et al., 2015), word sense disambiguation (Hovy and Hovy, 2012), temporal relation extraction (Ning et al., 2018), dependency parsing (Flannery et al., 2012), and named entity recognition (Jie et al., 2019).

In particular, Jie et al. (2019) study a similar problem with a few key differences: since they remove entity surfaces randomly, the dataset is too easy; and they do not use constraints on their output. We compare against their results in our experiments.

Our proposed method is most closely aligned with the Constraint Driven Learning (CoDL) framework (Chang et al., 2007), in which an iterative algorithm reminiscent of self-training is guided by constraints that are applied at each iteration.

3 Constrained Binary Learning

Our method assigns instance weights to all negative elements (tokens tagged as O), so that false negatives have low weights, and all other instances have high weights. We calculate weights according to the confidence predictions of a classifier trained iteratively over the partially annotated data. We refer to our method as Constrained Binary Learning (CBL).[2]

We will first describe the motivation for this approach before moving on to the mechanics. We start with partially annotated data (which we call set T) in which some, but not all, positives are annotated (set P), and no negative is labeled. By default, we assume that any instance not labeled as positive is labeled as negative as opposed to unlabeled. This data (set N) is noisy in the sense that many true positives are labeled as negative (these are *false negatives*). Clearly, training on T as-is will result in a noisy classifier.

Two possible approaches are: **1)** find the false negatives and label them correctly, or **2)** find the false negatives and remove them. The former method affords more training data, but runs the risk of adding noise, which could be worse than the original partial annotations. The latter is more forgiving because of an asymmetry in the penalties: it is important to remove all false negatives in N, but inadvertently removing true negatives from N is typically not a problem, especially in NER, where negative examples dominate. Further, a binary model (only two labels) is sufficient in this case, as we need only detect entities, not type them.

We choose the latter method, but instead of removing false negatives, we adopt an instance-weighting approach, in which each instance is assigned a weight $v_i \geq 0$ according to confidence in the labeling of that instance. A weight of 0 means that the loss this instance incurs during training will not update the model.

With this in mind, CBL takes two phases: first, it learns a binary classifier λ using a constrained iterative process modeled after the CODL framework (Chang et al., 2007), and depicted in Figure 2. The core of

[1] See also: https://hazyresearch.github.io/snorkel/blog/ws_blog_post.html

[2] Publication details (including code) can be found at cogcomp.org/page/publication_view/888

Require:
P : positive tokens
N : noisy negative tokens
C : constraints

1: $T = N \cup P$
2: $V \leftarrow$ Initialize T with weights (Optional)
3: **while** stopping condition not met **do**
4: $\quad \lambda \leftarrow \text{train}(T, V)$
5: $\quad \hat{T} \leftarrow \text{predict}(\lambda, T)$
6: $\quad T, V \leftarrow \text{inference}(\hat{T}, C)$
7: **end while**
8: return λ

Figure 2: Constrained Binary Learning (CBL) algorithm (phase 1). The core of the algorithm is in the while loop, which iterates over training on T, predicting on T and correcting those predictions.

the algorithm is the train-predict-infer loop. The training process (line 4) is weighted, using weights V. At the start, these can be all 1 (Raw), or can be initialized with prior knowledge. The learned model is then used to predict on all of T (line 5). In the inference step (line 6), we take the predictions from the prior round and the constraints C and produce a new labeling on T, and a new set of weights V. The details of this inference step are presented later in this section. Although our ultimate strategy is simply to assign weights (not change labels), in this inner loop, we update the labels on N according to classifier predictions.

In the second phase of CBL, we use the λ trained in the previous phase to assign weights to instances as follows:

$$v_i = \begin{cases} 1.0 & \text{if } x_i \in P \\ P_\lambda(y_i = \text{O} \mid x_i) & \text{if } x_i \in N \end{cases} \quad (1)$$

Where $P_\lambda(y_i = \text{O} \mid x_i)$ is understood as the classifier's confidence that instance x_i takes the negative label. In practice it is sufficient to use any confidence score from the classifier, not necessarily a probability. If the classifier has accurately learned to detect entities, then for all the false negatives in N, $P_\lambda(y_i = \text{O}|x_i)$ is small, which is the goal.

Ultimately, we send the original multiclass partially annotated dataset along with final weights V to a standard weighted NER classifier to learn a model. No weights are needed at test time.

3.1 NER with CBL

So far, we have given a high-level view of the algorithm. In this section, we will give more low-level details, especially as they relate to the specific problem of NER. One contribution of this work is the inference step (line 6), which we address using a constrained Integer Linear Program (ILP) and describe in this section. However, the constraints are based on a value we call the *entity ratio*. First, we describe the entity ratio, then we describe the constraints and stopping condition of the algorithm.

3.1.1 Entity ratio and Balancing

We have observed that NER datasets tend to hold a relatively stable ratio of entity tokens to total tokens. We refer to this ratio as b, and define it with respect to some labeled dataset as:

$$b = \frac{|P|}{|P| + |N|} \quad (2)$$

where N is the set of negative examples. Previous work has shown that in fully-annotated datasets the entity ratio tends to be about 0.09 ± 0.05, depending on the dataset and genre (Augenstein et al., 2017). Intuitively, knowledge of the gold entity ratio can help us estimate when we have found all the false negatives.

In our main experiments, we assume that the entity ratio with respect to the gold labeling is known for each training dataset. A similar assumption was made in Elkan and Noto (2008) when determining the c value, and in Grave (2014) in the constraint determining the percentage of OTHER examples. However, we also show in Section 4.8 that knowledge of this ratio is not strictly necessary, and a flat value across all datasets produces similar performance.

With a weighted training set, it is also useful to define the weighted entity ratio.

$$b = \frac{|P|}{|P| + \sum_{i \in N} v_i} \quad (3)$$

When training an NER model on weighted data, one can change the weighted entity ratio to achieve different effects. To make balanced predictions on test, the entity ratio in the training data should roughly match that of the test data (Chawla, 2005). To bias a model towards predicting positives or predicting negatives, the weighted entity ratio can be set higher or lower respectively. This effect is pronounced when using linear methods for NER, but not as clear in neural methods.

To change the entity ratio, we scale the weights in N by a scaling constant γ. Targeting a particular b^*, we may write:

$$b^* = \frac{|P|}{|P| + \gamma \sum_{i \in N} v_i} \quad (4)$$

We can solve for γ:

$$\gamma = \frac{(1 - b^*)|P|}{b^* \sum_{i \in N} v_i} \quad (5)$$

To obtain weights, v_i^*, that attain the desired entity ratio, b^*, we scale all weights in N by γ.

$$v_i^* = \gamma v_i \quad (6)$$

In the train-predict-infer loop, we balance the weights to a value near the gold ratio before training.

3.1.2 Constraints and Stopping Condition

We encode our constraints with an Integer Linear Program (ILP), shown in Figure 3. Intuitively, the job of the inference step is to take predictions ($\hat{T}$) and use knowledge of the task to 'fix' them.

In the objective function (Eqn. 8), token i is represented by two indicator variables y_{0i} and y_{1i}, representing negative and positive labels, respectively. Associated prediction scores C_0 and C_1 are from the classifier λ in the last round of predictions. The first constraint (Eqn. 9) encodes the fact that an instance cannot be both an entity and a non-entity.

The second constraint (Eqn. 10) enforces the ratio of positive to total tokens in the corpus to match a required entity ratio. $|T|$ is the total number of tokens in the corpus. b is the required entity ratio, which increases at each iteration. δ allows some flexibility, but is small.

Constraint 11 encodes that instances in P should be labeled positive since they were manually labeled and are by definition trustworthy. We set $\xi \geq 0.99$.

This framework is flexible in that more complex language- or task-specific constraints could be added. For example, in English and many other languages with Latin script, it may help to add a capitalization constraint. In languages with rich morphology, certain suffixes may indicate or contraindicate a named entity. For simplicity, and because of the number of languages in our experiments, we use only a few constraints.

After the ILP has selected predictions, we assign weights to each instance in preparation for training the next round. The decision process for an instance is:

$$v_i = \begin{cases} 1.0 & \text{If ILP labeled } x_i \text{ positive} \\ P_\lambda(y_i = \text{O} \mid x_i) & \text{Otherwise} \end{cases} \tag{7}$$

This is similar to Equation (1), except that the set of tokens that the ILP labeled as positive is larger than P. With new labels and weights, we start the next iteration.

The stopping condition for the algorithm is related to the entity ratio. One important constraint (Eqn. 10) governs how many positives are labeled at each round. This number starts at $|P|$ and is increased by a small value[3] at each iteration, thereby improving recall. Positive instances are chosen in two ways. First, all instances in P are constrained to be labeled positive (Eqn. 11). Second, the objective function ensures that high-confidence positives will be chosen. The stopping condition is met when the number of required positive instances (computed using gold unweighted entity ratio) equals the number of predicted positive instances.

4 Experiments

We measure the performance of our method on 8 different languages using artificially perturbed labels to simulate the partial annotation setting.

[3]The size of this value is related to how much we trust the ranking induced by prediction confidences. If we believed the ranking was perfect, we could take as many positives as we wanted and be finished in one round.

$$\max_{\mathbf{y}} \quad \sum_{i}^{|T|} C_{0i} y_{0i} + C_{1i} y_{1i} \tag{8}$$

$$\text{s.t.} \quad \forall i,\ y_{0i} + y_{1i} = 1 \tag{9}$$

$$b - \delta \leq \sum_i y_{1i}/|T| \leq b + \delta \tag{10}$$

$$\forall i,\ x_i \in P,\ \sum_i y_{1i} \geq \xi |P|, \tag{11}$$

Figure 3: ILP for the inference step

4.1 Data

We experiment on 8 languages. Four languages – English, German, Spanish, Dutch – come from the CoNLL 2002/2003 shared tasks (Tjong Kim Sang and De Meulder, 2003a,b). These are taken from newswire text, and have labelset of Person, Organization, Location, Miscellaneous.

The remaining four languages come from the LORELEI project (Strassel and Tracey, 2016). These languages are: Amharic (amh: LDC2016E87), Arabic (ara: LDC2016E89), Hindi (hin: LDC2017E62), and Somali (som: LDC2016E91). These come from a variety of sources including discussion forums, newswire, and social media. The labelset is Person, Organization, Location, Geo-political entity. We define train/development/test splits, taking care to keep a similar distribution of genres in each split. Data statistics for all languages are shown in Table 1.

4.2 Artificial Perturbation

We create partial annotations by perturbing gold annotated data in two ways: lowering recall (to simulate missing entities), and lowering precision (to simulate noisy annotations).

To lower recall, we replace gold named entity tags with O tags (for non-name). We do this by grouping named entity surface forms, and replacing tags on all occurrences of a randomly selected surface form until the desired amount remains. For example, if the token 'Bangor' is chosen to be untagged, then every occurrence of 'Bangor' will be untagged. We chose this slightly complicated method because the simplest idea (remove mentions randomly) leaves an artificially large diversity of surface forms, which makes the problem of discovering noisy entities easier.

To lower precision, we tag a random span (of a random start position, and a random length between 1 and 3) with a random named entity tag. We continue this process until we reach the desired precision. When both precision and recall are to be perturbed, the recall adjustment is made first, and then the number of random spans to be added is calculated by the entities that are left.

	Train			Test		
Lang.	b (%)	Tag	Tok	b (%)	Tag	Tok
English	16.6	23K	203K	17.3	5K	46K
Spanish	12.3	18K	264K	11.9	3K	51K
German	8.0	11K	206K	9.9	3K	51K
Dutch	9.5	13K	202K	8.3	4K	68K
Amharic	11.2	3K	52K	11.3	1K	18K
Arabic	12.6	4K	60K	10.2	931	16K
Hindi	7.38	4K	74K	7.53	1K	25K
Somali	11.2	4K	57K	11.9	1K	16K

Table 1: Data statistics for all languages, showing number of tags and tokens in Train and Test. The tag counts represent individual spans, not tokens. That is, "[Barack Obama]$_{\text{PER}}$" counts as one tag, not two. The b column shows the entity ratio as a percentage.

4.3 NER Models

In principle, CBL can use any NER method that can be trained with instance weights. We experiment with both non-neural and neural models.

4.3.1 Non-neural Model

For our non-neural system, we use a version of Cogcomp NER (Ratinov and Roth, 2009; Khashabi et al., 2018) modified to use Weighted Averaged Perceptron. This operates on a weighted training set $D_w = \{(x_i, y_i, v_i)\}_{i=1}^{N}$, where N is the number of training examples, and $v_i \geq 0$ is the weight on the ith training example. In this non-neural system, a training example is a word with context encoded in the features. We change only the update rule, where the learning rate α is multiplied by the weight:

$$\mathbf{w} = \mathbf{w} + \alpha v_i y_i (\mathbf{w}^T x_i) \tag{12}$$

We use a standard set of features, as documented in Ratinov and Roth (2009). In order to keep the language-specific resources to a minimum, we did not use any gazetteers for any language.[4] One of the most important features is Brown clusters, trained for 100, 500, and 1000 clusters for the CoNLL languages, and 2000 clusters for the remaining languages. We trained these clusters on Wikipedia text for the four CoNLL languages, and on the same monolingual text used to train the word vectors (described in Section 4.3.2).

4.3.2 Neural Model

A common neural model for NER is the BiLSTM-CRF model (Ma and Hovy, 2016). However, because the Conditional Random Field (CRF) layer calculates loss at the sentence level, we need a different method to incorporate token weights. We use a variant of the CRF that allows partial annotations by marginalizing over all possible sequences (Tsuboi et al., 2008).

[4]Separate experiments show that omitting gazetteers impacts performance only slightly.

When using a standard BiLSTM-CRF model, the loss of a dataset (D) composed of sentences (s) is calculated as:

$$\mathcal{L} = -\sum_{s \in D} \log P_\theta(\mathbf{y}^{(s)} | \mathbf{x}^{(s)}) \tag{13}$$

Where $P_\theta(\mathbf{y}^{(s)} | \mathbf{x}^{(s)})$ is calculated by the CRF over outputs from the BiLSTM. In the marginal CRF framework, it is assumed that $\mathbf{y}^{(s)}$ is necessarily partial, denoted as $\mathbf{y}_p^{(s)}$. To incorporate partial annotations, the loss is calculated by marginalizing over all possible sequences consistent with the partial annotations, denoted as $C(\mathbf{y}_p^s)$.

$$\mathcal{L} = -\sum_{s \in D} \log \sum_{\mathbf{y} \in C(\mathbf{y}_p^{(s)})} P_\theta(\mathbf{y} | \mathbf{x}^{(s)}) \tag{14}$$

However, this formulation assumes that all possible sequences are equally likely. To address this, Jie et al. (2019) introduced a way to weigh sequences.

$$\mathcal{L} = -\sum_{s \in D} \log \sum_{\mathbf{y} \in C(\mathbf{y}_p^{(s)})} q(\mathbf{y} | \mathbf{x}^{(s)}) P_\theta(\mathbf{y} | \mathbf{x}^{(s)}) \tag{15}$$

It's easy to see that this formulation is a generalization of the standard CRF if $q(.) = 1$ for the gold sequence $\mathbf{y}$, and 0 for all others.

The product inside the summation depends on tag transition probabilities and tag emission probabilities, as well as token-level "weights" over the tagset. These weights can be seen as defining a soft gold labeling for each token, corresponding to confidence in each label.

For clarity, define the soft gold labeling over each token x_i as $\mathbf{G}_i \in [0, 1]^L$, where L is the size of the labelset. Now, we may define $q(.)$ as:

$$q(\mathbf{y} | \mathbf{x}^{(s)}) = \prod_i G_i^{y_i}$$

Where $G_i^{y_i}$ is understood as the weight in $\mathbf{G}_i$ that corresponds to the label y_i.

We incorporate our instance weights in this model with the following intuitions. Recall that if an instance weight $v_i = 0$, this indicates low confidence in the label on token x_i, and therefore the labeling should not update the model at training time. Conversely, if $v_i = 1$, then this label is to be trusted entirely.

If $v_i = 0$, we set the soft labeling weights over x_i to be uniform, which is as good as no information. Since v_i is defined as confidence in the O label, the soft labeling weight for O increases proportionally to v_i. Any remaining probability mass is distributed evenly among the other labels.

To be precise, for tokens in N, we calculate values for $\mathbf{G}_i$ as follows:

$$G_i^O = \max(1/L, v_i)$$

$$G_i^{\text{non-}O} = \frac{1 - G_i^O}{L - 1}$$

For example, consider phase 1 of Constrained Binary Learning, in which the labelset is collapsed to two labels ($L = 2$). Assuming that the O label has index 0, then if $v_i = 0$, then $\mathbf{G}_i = [0.5, 0.5]$. If $v_i = 0.6$, then $\mathbf{G}_i = [0.6, 0.4]$.

For tokens in P (which have some entity label with high confidence), we always set $\mathbf{G}_i$ with 1 in the given label index, and 0 elsewhere.

We use pretrained GloVe (Pennington et al., 2014) word vectors for English, and the same pretrained vectors used in Lample et al. (2016) for Dutch, German, and Spanish. The other languages are distributed with monolingual text (Strassel and Tracey, 2016), which we used to train our own skip-n-gram vectors.

4.4 Baselines

We compare against several baselines, including two from prior work.

4.4.1 Raw annotations

The simplest baseline is to do nothing to the partially annotated data and train on it as is.

4.4.2 Instance Weights

Although CBL works with no initialization (that is, all tokens with weight 1), we found that a good weighting scheme can boost performance for certain models. We design weighting schemes that give instances in N weights corresponding to an estimate of the label confidence.[5] For example, non-name tokens such as *respectfully* should have weight 1, but possible names, such as *Russell*, should have a low weight, or 0. We propose two weighting schemes: frequency-based and window-based.

For the frequency-based weighting scheme, we observed that names have relatively low frequency (for example, *Kennebunkport*, *Dushanbe*) and common words are rarely names (for example *the*, *and*, *so*). We weigh each instance in N according to its frequency.

$$v_i^{\text{freq}} = freq(x_i) \tag{16}$$

where $freq(x_i)$ is the frequency of the i^{th} token in N divided by the count of the most frequent token. In our experiments, we computed frequencies over $P+N$, but these could be estimated on any sufficiently large corpus. We found that the neural model performed poorly when the weights followed a Zipfian distribution (e.g. most weights very small), so for those experiments, we took the log of the token count before normalizing.

For the window-based weighting scheme, noting that names rarely appear immediately adjacent to each other in English text, we set weights for tokens within a window of size 1 of a name (identified in P) to be 1.0, and for tokens farther away to be 0.

$$v_i^{\text{window}} = \begin{cases} 1.0 & \text{if } d_i \leq 1 \\ 0.0 & \text{otherwise} \end{cases} \tag{17}$$

where d_i is the distance of the i^{th} token to the nearest named entity in P.

Finally, we combine the two weighting schemes as:

$$v_i^{\text{combined}} = \begin{cases} 1.0 & \text{if } d_i \leq 1 \\ v_i^{\text{freq}} & \text{otherwise} \end{cases} \tag{18}$$

4.4.3 Self-training with Marginal CRF

Jie et al. (2019) propose a model based on marginal CRF (Tsuboi et al., 2008) (described in Section 4.3.2). They follow a self-training framework with cross-validation, using the trained model over all but one fold to update gold labeling distributions in the final fold. This process continues until convergence. They use a partial-CRF framework similar to ours, but taking predictions at face value, without constraints.

4.4.4 Neural Network with Noise Adaptation

Following Hedderich and Klakow (2018), we used a neural network with a noise adaptation layer.[6] This extra layer attempts to correct noisy examples given a probabilistic confusion matrix of label noise. Since this method needs a small amount of labeled data, we selected 500 random tokens to be the gold training set, in addition to the partial annotations.

As with our BiLSTM experiments, we use pretrained GloVe word vectors for English, and the same pretrained vectors used in Lample et al. (2016) for Dutch, German, and Spanish. We omit results from the remaining languages because the scores were substantially worse even than training on raw annotations.

4.5 Experimental Setup and Results

We show results from our experiments in Table 2. In all experiments, the training data is perturbed at 90% precision and 50% recall. These parameters are similar to the scores obtained by human annotators in a foreign language (see Section 5). We evaluate each experiment with both non-neural and neural methods.

First, to get an idea of the difficulty of NER in each language, we report scores from models trained on gold data without perturbation (*Gold*). Then we report results from an Oracle Weighting scheme (*Oracle Weighting*) that takes partially annotated data and assigns weights with knowledge of the true labels. Specifically, mislabeled entities in set N are given weight 0, and all other tokens are given weight 1.0. This scheme is free from labeling noise, but should still get lower scores than Gold because of the smaller number of entities. Since our method estimates these weights, we do not expect CBL to outperform the Oracle method. Next, we show results from all baselines. The bottom two sections are our results, first with no initialization (*Raw*), and CBL over that, then with *Combined Weighting* initialization, and CBL over that.

[5] All elements of P always have weight 1

[6] The code was kindly provided by the authors.

Method \ Language	Tool	**eng**	**deu**	**esp**	**ned**	**amh**	**ara**	**hin**	**som**	avg
Gold	Cogcomp	89.1	72.5	82.5	82.6	67.2	53.4	74.4	80.3	75.3
	BiLSTM-CRF	90.3	77.3	85.2	81.1	69.2	52.8	73.8	82.3	76.5
Oracle Weighting	Cogcomp	83.7	65.7	76.2	76.4	54.3	42.0	56.3	68.5	65.4
	BiLSTM-CRF	87.8	70.2	78.5	70.4	60.4	43.4	57.6	73.2	67.7
Noise Adaptation (Hedderich, 2018)		61.5	46.1	57.3	41.5	–	–	–	–	–
Self-training (Jie et al., 2019)		82.3	65.2	76.3	65.5	52.1	40.1	55.1	65.3	62.7
Raw Annotations	Cogcomp	54.8	36.9	49.5	47.9	31.0	32.6	30.9	44.0	40.9
	BiLSTM-CRF	73.3	57.7	61.9	58.3	42.2	36.8	47.5	54.9	54.1
CBL-Raw	CogComp	74.7	63.0	68.7	67.0	45.0	37.8	50.6	67.9	59.3
	BiLSTM-CRF	**84.6**	**67.9**	**79.6**	70.0	**52.9**	42.1	55.2	**70.4**	**65.3**
Combined Weighting	Cogcomp	75.2	56.6	70.8	70.8	46.5	**44.1**	57.5	60.2	60.2
	BiLSTM-CRF	73.5	60.3	64.9	61.9	48.0	38.0	49.0	56.6	56.5
CBL-Combined	Cogcomp	77.3	61.8	74.0	**72.4**	49.2	43.7	**58.2**	67.6	63.0
	BiLSTM-CRF	81.1	64.9	74.9	63.4	52.2	39.8	52.0	67.0	61.9

Table 2: F1 scores on English, German, Spanish, Dutch, Amharic, Arabic, Hindi, and Somali. Each section shows performance of both Cogcomp (non-neural) and BiLSTM (neural) systems. *Gold* is using all available gold training data to train. *Oracle Weighting* uses full entity knowledge to set weights on N. The next section shows prior work, followed by our methods. The column to the farthest right shows the average score over all languages. Bold values are the highest per column. On average, our best results are found in the uninitialized (*Raw*) CBL from BiLSTM-CRF.

4.6 Analysis

Regardless of initialization or model, CBL improves over the baselines. Our best model, *CBL-Raw BiLSTM-CRF*, improves over the *Raw Annotations BiLSTM-CRF* baseline by 11.2 points F1, and the *Self-training* prior work by 2.6 points F1, showing that it is an effective way to address the problem of partial annotation. Further, the best CBL version for each model is within 3 points of the corresponding *Oracle* ceiling, suggesting that this weighting framework is nearly saturated.

The *Combined* weighting scheme is surprisingly effective for the non-neural model, which suggests that the intuition about frequency as distinction between names and non-names holds true. It gives modest improvement in the neural model. The *Self-training* method is effective, but is outperformed by our best CBL method, a difference we discuss in more detail in Section 4.7. The *Noise Adaptation* method outperforms the *Raw annotations Cogcomp* baseline in most cases, but does not reach the performance of the *Self-training* method, despite using some fully labeled data.

It is instructive to compare the neural and non-neural versions of each setup. The neural method is better overall, but is less able to learn from the knowledge-based initialization weights. In the non-neural method, the difference between *Raw* and *Combined* is nearly 20 points, but the difference in the neural model is less than 3 points. *Combined* versions of the non-neural method outperform the neural method on 3 languages: Dutch, Arabic, and Hindi. Further, in the neural method, *CBL-Raw* is always worse than *CBL-Combined*. This may be due to the way that weights are used in each model. In the non-neural model, a low enough weight completely cancels the token, whereas in the neural model it is still used in training. Since the neural model performs well in the *Oracle* setting, we know that it can learn from hard weights, but it may have trouble with the subtle differences encoded in frequencies. We leave it to future work to discover improved ways of incorporating instance weights in a BiLSTM-CRF.

In seeking to understand the details of the other results, we need to consider the precision/recall tradeoff. First, all scores in the *Gold* row had higher precision than recall. Then, training on raw partially annotated data biases a classifier strongly towards predicting few entities. All results from the *Raw annotations* row have precision more than double the recall (e.g. Dutch Precision, Recall, F1 were: 91.5, 32.4, 47.9). In this context, the problem this paper explores is how to improve the recall of these datasets without harming the precision.

4.7 Difference from Prior Work

While our method has several superficial similarities with prior work, most notably Jie et al. (2019), there are some crucial differences.

Our methods are similar in that they both use a model trained at each step to assign a soft gold-labeling to each token. Each algorithm iteratively trains models using weights from the previous steps.

One difference is that Jie et al. (2019) use cross-validation to train, while we follow Chang et al. (2007) and retrain with the entire training set at each round.

	Avg F1		
Method \ b	10%	15%	Gold
Oracle Weighting	65.8	**65.9**	65.4
Raw annotations	40.9	40.9	40.9
Combined Weighting	59.9	**60.2**	**60.2**
CBL-Combined	62.4	62.3	**63.0**

Table 3: Experimenting with different entity ratios. Scores reported are average F1 across all languages. *Gold* b value refers to using the gold annotated data to calculate the optimal entity ratio. This table shows that exact knowledge of the entity ratio is not required for CBL to succeed.

However, the main difference has to do with the focus of each algorithm. Recall the discussion in Section 3 regarding the two possible approaches of **1**) find the false negatives and label them correctly, and **2**) find the false negatives and remove them. Conceptually, the former was the approach taken by Jie et al. (2019), the latter was our approach. Another way to look at this is as focusing on predicting correct tag labels ((Jie et al., 2019)) or focus on predicting O tags with high confidence (ours).

Even though they use soft labeling (which they show to be consistently better than hard labeling), it is possible that the predicted tag distribution is incorrect. Our approach allows us to avoid much of the inevitable noise that comes from labelling with a weak model.

4.8 Varying the Entity Ratio

Recall that the entity ratio is used for balancing and for the stopping criteria in CBL. In all our experiments so far, we have used the gold entity ratio for each language, as shown in Table 1. However, exact knowledge of entity ratio is unlikely in the absence of gold data. Thus, we experimented with selecting a default b value, and using it across all languages, with the Cogcomp model. We chose values of 10% and 15%, and report F1 averaged across all languages in Table 3.

While the gold b value is the best for *CBL-Combined*, the flat 15% ratio is best for all other methods, showing that exact knowledge of the entity ratio is not necessary.

5 Bengali Case Study

So far our experiments have shown effectiveness on artificially perturbed labels, but one might argue that these systematic perturbations don't accurately simulate real-world noise. In this section, we show how our methods work in a real-world scenario, using Bengali data partially labeled by non-speakers.

5.1 Non-speaker Annotations

In order to compare with prior work, we used the train/test split from Zhang et al. (2016). We removed all gold labels from the train split, romanized it[7] (Hermjakob et al., 2018), and presented it to two non-Bengali speaking annotators using the TALEN interface (Mayhew and Roth, 2018). The instructions were to move quickly and annotate names only when there is high confidence (e.g. when you can also identify the English version of the name). They spent about 5 total hours annotating, without using Google Translate. This sort of non-speaker annotation is possible because the text contains many 'easy' entities – foreign names – which are noticeably distinct from native Bengali words. For example, consider the following:

Num tokens	49K
Num sentences	2435
Num name tokens	2326
Entity ratio	4.66%
Num unique name tokens	664
Annotator 1 Prec/Rec/F1	84/34/48
Annotator 2 Prec/Rec/F1	79/28/42
Combined Prec/Rec/F1	83/32/47

Table 4: Bengali Data Statistics. The P/R/F1 scores are computed for the non-speaker annotator with respect to the gold training data.

- **Romanized Bengali**: ebisi'ra giliyyaana phinnddale aaja pyaalestaaina adhiinastha gaajaa theke aaja raate ekhabara jaaniyyechhena .
- **Translation**[8]: ABC's Gillian Fondley has reported today from Gaza under Palestine today.

The entities are Gillian Findlay, ABC, Palestine, and Gaza. While a fast-moving annotator may not catch most of these, 'pyaalestaaina' could be considered an 'easy' entity, because of its visual and aural similarity to 'Palestine.' A clever annotator may also infer that if Palestine is mentioned, then Gaza may be present.

Annotators are moving fast and being intentionally non-thorough, so the recall will be low. Since they do not speak Bengali, there are likely to be some mistakes, so the precision may drop slightly also. This is exactly the noisy partial annotation scenario addressed in this paper. The statistics of this data can be seen in Table 4, including annotation scores computed with respect to the gold training data for each annotator, as well as the combined score.

We show results in Table 5, using the BiLSTM-CRF model. We compare against other low-resource approaches published on this dataset, including two based on Wikipedia (Tsai et al., 2016; Pan et al., 2017), another based on lexicon translation from a high-resource language (Mayhew et al., 2017). These prior methods operate under somewhat different paradigms than

[7]This step is vitally important. We used `www.isi.edu/~ulf/uroman.html`

[8]From `translate.google.com`

	Test		
Scheme	P	R	F1
(Zhang et al., 2016)	-	-	34.8
(Tsai et al., 2016)	-	-	43.3
(Pan et al., 2017)	-	-	44.0
(Mayhew et al., 2017)	-	-	46.2
BILSTM-CRF			
Train on Gold	71.6	70.2	70.9
Raw annotations	**73.0**	23.8	35.9
Combined Weighting	65.9	34.2	45.0
CBL-Raw	57.8	**47.3**	**52.0**
CBL-Combined	58.3	44.2	50.2

Table 5: Bengali manual annotation results. Our methods improve on state of the art scores by over 5 points F1 given a relatively small amount of noisy and incomplete annotations from non-speakers.

this work, but have the same goal: maximizing performance in the absence of gold training data.

Raw annotations is defined as before, and gives similar high-precision low-recall results. The *Combined Weighting* scheme improves over Raw annotations by 10 points, achieving a score comparable to the prior state of the art. Beyond that, *CBL-Raw* outperforms the prior best by nearly 6 points F1, although *CBL-Combined* again underwhelms.

To the best of our knowledge, this is the first result showing a method for non-speaker annotations to produce high-quality NER scores. The simplicity of this method and the small time investment for these results gives us confidence that this method can be effective for many low-resource languages.

6 Conclusions

We explore an understudied data scenario, and introduce a new constrained iterative algorithm to solve it. This algorithm performs well in experimental trials in several languages, on both artificially perturbed data, and in a truly low-resource situation.

7 Acknowledgements

This work was supported by Contracts HR0011-15-C-0113 and HR0011-18-2-0052 with the US Defense Advanced Research Projects Agency (DARPA). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

Isabelle Augenstein, Leon Derczynski, and Kalina Bontcheva. 2017. Generalisation in named entity recognition: A quantitative analysis. *Computer Speech and Language*, 44:61–83.

M. Chang, L. Ratinov, and D. Roth. 2007. Guiding semi-supervision with constraint-driven learning. In *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 280–287, Prague, Czech Republic. Association for Computational Linguistics.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542.

Nitesh V. Chawla. 2005. Data mining for imbalanced datasets: An overview. In *The Data Mining and Knowledge Discovery Handbook*.

Mostafa Dehghani, Arash Mehrjou, Stephan Gouws, Jaap Kamps, and Bernhard Schölkopf. 2017. Fidelity-weighted learning. *CoRR*, abs/1711.02799.

Charles Elkan and Keith Noto. 2008. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220. ACM.

Eraldo Rezende Fernandes and Ulf Brefeld. 2011. Learning from partially annotated sequences. In *ECML/PKDD*.

Daniel Flannery, Yusuke Miyao, Graham Neubig, and Shinsuke Mori. 2012. A pointwise approach to training dependency parsers from partially annotated corpora. *Information and Media Technologies*, 7(4):1489–1513.

Jason A. Fries, Sen Wu, Alexander Ratner, and Christopher Ré. 2017. Swellshark: A generative model for biomedical named entity recognition without labeled data. *CoRR*, abs/1704.06360.

Jacob Goldberger and Ehud Ben-Reuven. 2017. Training deep neural-networks using a noise adaptation layer. In *ICLR*.

Edouard Grave. 2014. Weakly supervised named entity classification. In *AKBC*.

Michael A. Hedderich and Dietrich Klakow. 2018. Training a neural network in a low-resource setting on automatically annotated noisy data. In *Proceedings of the Workshop on Deep Learning Approaches for Low-Resource NLP*, pages 12–18, Melbourne. Association for Computational Linguistics.

Ulf Hermjakob, Jonathan May, and Kevin Knight. 2018. Out-of-the-box universal Romanization tool uroman. In *Proceedings of ACL 2018, System Demonstrations*, pages 13–18, Melbourne, Australia. Association for Computational Linguistics.

Jerónimo Hernández-González, Inaki Inza, and Jose A Lozano. 2016. Weak supervision and other non-standard classification problems: a taxonomy. *Pattern Recognition Letters*, 69:49–55.

Dirk Hovy and Eduard Hovy. 2012. Exploiting partial annotations with EM training. In *Proceedings of the NAACL-HLT Workshop on the Induction of Linguistic Structure*, pages 31–38, Montréal, Canada. Association for Computational Linguistics.

Zhanming Jie, Pengjun Xie, Wei Lu, Ruixue Ding, and Linlin Li. 2019. Better modeling of incomplete annotations for named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 729–734, Minneapolis, Minnesota. Association for Computational Linguistics.

Daniel Khashabi, Mark Sammons, Ben Zhou, Tom Redman, Christos Christodoulopoulos, Vivek Srikumar, Nicholas Rizzolo, Lev Ratinov, Guanheng Luo, Quang Do, Chen-Tse Tsai, Subhro Roy, Stephen Mayhew, Zhili Feng, John Wieting, Xiaodong Yu, Yangqiu Song, Shashank Gupta, Shyam Upadhyay, Naveen Arivazhagan, Qiang Ning, Shaoshi Ling, and Dan Roth. 2018. Cogcompnlp: Your swiss army knife for nlp. In *LREC*.

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

Wee Sun Lee and Bing Liu. 2003. Learning with positive and unlabeled examples using weighted logistic regression. In *ICML*.

Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and Philip S. Yu. 2003. Building text classifiers using positive and unlabeled examples. In *ICDM*.

Bing Liu, Wee Sun Lee, Philip S. Yu, and Xiaoli Li. 2002. Partially supervised classification of text documents. In *ICML*.

Xuezhe Ma and Eduard 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 (Volume 1: Long Papers)*, pages 1064–1074, Berlin, Germany. Association for Computational Linguistics.

Stephen Mayhew and Dan Roth. 2018. TALEN: Tool for annotation of low-resource ENtities. In *Proceedings of ACL 2018, System Demonstrations*, pages 80–86, Melbourne, Australia. Association for Computational Linguistics.

Stephen Mayhew, Chen-Tse Tsai, and Dan Roth. 2017. Cheap translation for cross-lingual named entity recognition. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Shinsuke Mori, Yosuke Nakata, Graham Neubig, and Tetsuro Sasada. 2015. Pointwise prediction and sequence-based reranking for adaptable part-of-speech tagging. In *PACLING*.

Qiang Ning, Zhongzhi Yu, Chuchu Fan, and Dan Roth. 2018. Exploiting partially annotated data in temporal relation extraction. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pages 148–153, New Orleans, Louisiana. Association for Computational Linguistics.

Joel Nothman, James R Curran, and Tara Murphy. 2008. Transforming wikipedia into named entity training data. In *Proceedings of the Australian Language Technology Workshop*, pages 124–132.

Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. 2013. Learning multilingual named entity recognition from wikipedia. *Artif. Intell.*, 194:151–175.

Xiaoman Pan, Boliang Zhang, Jonathan May, Joel Nothman, Kevin Knight, and Heng Ji. 2017. Cross-lingual name tagging and linking for 282 languages. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1946–1958, Vancouver, Canada. Association for Computational Linguistics.

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

L. Ratinov and D. Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*.

Alexander Ratner, Stephen H. Bach, Henry R. Ehrenberg, Jason Alan Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, 11 3:269–282.

Dan Roth. 2017. Incidental supervision: Moving beyond supervised learning. In *Proc. of the Conference on Artificial Intelligence (AAAI)*.

Stephanie Strassel and Jennifer Tracey. 2016. LORELEI language packs: Data, tools, and resources for technology development in low resource languages. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 3273–3280, Portorož, Slovenia. European Language Resources Association (ELRA).

Erik F. Tjong Kim Sang and Fien De Meulder. 2003a. 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.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003b. 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.

Chen-Tse Tsai, Stephen Mayhew, and Dan Roth. 2016. Cross-lingual named entity recognition via wikification. In *Proc. of the Conference on Computational Natural Language Learning (CoNLL)*.

Yuta Tsuboi, Hisashi Kashima, Hiroki Oda, Shinsuke Mori, and Yuji Matsumoto. 2008. Training conditional random fields using incomplete annotations. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 897–904. Association for Computational Linguistics.

Boliang Zhang, Xiaoman Pan, Tianlu Wang, Ashish Vaswani, Heng Ji, Kevin Knight, and Daniel Marcu. 2016. Name tagging for low-resource incident languages based on expectation-driven learning. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 249–259, San Diego, California. Association for Computational Linguistics.