SeqMix: Augmenting Active Sequence Labeling via Sequence Mixup

Rongzhi Zhang
Georgia Tech
rongzhi.zhang@gatech.edu

Yue Yu
Georgia Tech
yueyu@gatech.edu

Chao Zhang
Georgia Tech
chaozhang@gatech.edu

Abstract

Active learning is an important technique for low-resource sequence labeling tasks. However, current active sequence labeling methods use the queried samples alone in each iteration, which is an inefficient way of leveraging human annotations. We propose a simple but effective data augmentation method to improve label efficiency of active sequence labeling. Our method, SeqMix, simply augments the queried samples by generating extra labeled sequences in each iteration. The key difficulty is to generate plausible sequences along with token-level labels. In SeqMix, we address this challenge by performing mixup for both sequences and token-level labels of the queried samples. Furthermore, we design a discriminator during sequence mixup, which judges whether the generated sequences are plausible or not. Our experiments on Named Entity Recognition and Event Detection tasks show that SeqMix can improve the standard active sequence labeling method by 2.27%–3.75% in terms of F_1 scores. The code and data for SeqMix can be found at `https://github.com/rz-zhang/SeqMix`.

1 Introduction

Many NLP tasks can be formulated as sequence labeling problems, such as part-of-speech (POS) tagging (Zheng et al., 2013), named entity recognition (NER) (Lample et al., 2016), and event extraction (Yang et al., 2019). Recently, neural sequential models (Lample et al., 2016; Akbik et al., 2018; Vaswani et al., 2017) have shown strong performance for various sequence labeling task. However, these deep neural models are *label hungry*—they require large amounts of annotated sequences to achieve strong performance. Obtaining large amounts of annotated data can be too expensive for practical sequence labeling tasks, due to token-level annotation efforts.

Active learning is an important technique for sequence labeling in low-resource settings. Active sequence labeling is an iterative process. In each iteration, a fixed number of unlabeled sequences are selected by a query policy for annotation and then model updating, in hope of maximally improving model performance. For example, Tomanek et al. (2007); Shen et al. (2017) select query samples based on data uncertainties; Hazra et al. (2019) compute model-aware similarity to eliminate redundant examples and improve the diversity of query samples; and Fang et al. (2017); Liu et al. (2018) use reinforcement learning to learn query policies. However, existing methods for active sequence labeling all use the queried samples *alone* in each iteration. We argue that the queried samples provide limited data diversity, and using them alone for model updating is inefficient in terms of leveraging human annotation efforts.

We study the problem of enhancing active sequence labeling via data augmentation. We aim to generate augmented labeled sequences for the queried samples in each iteration, thereby introducing more data diversity and improve model generalization. However, data augmentation for active sequence labeling is challenging, because we need to generate sentences and token-level labels jointly. Prevailing generative models (Zhang et al., 2016; Bowman et al., 2016) are inapplicable because they can only generate word sequences without labels. It is also infeasible to apply heuristic data augmentation methods such as context-based words substitution (Kobayashi, 2018), synonym replacement, random insertion, swap, and deletion (Wei and Zou, 2019), paraphrasing (Cho et al., 2019) or back translation (Xie et al., 2019), because label composition is complex for sequence labeling. Directly using these techniques to manipulate tokens may inject incorrectly labeled sequences into training data and harm model performance.

We propose SeqMix, a data augmentation method for generating sub-sequences along with their labels based on *mixup* (Zhang et al., 2018). Under the active sequence labeling framework, SeqMix is capable of generating plausible pseudo labeled sequences for the queried samples in each iteration. This is enabled by two key techniques in SeqMix: (1) First, in each iteration, it searches for pairs of eligible sequences and mixes them both in the feature space and the label space. (2) Second, it has a discriminator to judge if the generated sequence is plausible or not. The discriminator is designed to compute the perplexity scores for all the generated candidate sequences and select the low-perplexity sequences as plausible ones.

We show that SeqMix consistently outperforms standard active sequence labeling baselines under different data usage percentiles with experiments on Named Entity Recognition and Event Detection tasks. On average, it achieves $2.95\%, 2.27\%, 3.75\%$ F_1 improvements on the CoNLL-2003, ACE05 and WebPage datasets. The advantage of SeqMix is especially prominent in low-resource scenarios, achieving $12.06\%, 8.86\%, 16.49\%$ F_1 improvements to the original active learning approach on the above three datasets. Our results also verify the proposed mixup strategies and the discriminator are vital to the performance of SeqMix.

2 Preliminaries

2.1 Problem Definition

Many NLP problems can be formulated as sequence labeling problems. Given an input sequence, the task is to annotate it with token-level labels. The labels often consist of a position prefix provided by a labeling schema and a type indicator provided by the specific task. For example, in the named entity recognition task, we can adopt the `BIO` (Beginning, Inside, Outside) tagging scheme (Màrquez et al., 2005) to assign labels for each token: the first token of an entity mention with type `X` is labeled as `B-X`, the tokens inside that mention are labeled as `I-X` and the non-entity tokens are labeled as `O`.

Consider a large unlabeled corpus $\mathcal{U}$, traditional active learning starts from a small annotated seed set $\mathcal{L}$, and utilizes a query function $\psi(\mathcal{U}, K, \gamma(\cdot))$ to obtain K most informative unlabeled samples $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_K\}$ along with their labels $\mathcal{Y} = \{y_1, \cdots, y_K\}$, where $\gamma(\cdot)$ is the query policy. Then, we remove $\mathcal{X}$ from the unlabeled data $\mathcal{U}$ and repeat the above procedure until the satisfactory performance achieved or the annotation capacity reached.

In SeqMix, we aim to further exploit the annotated set $\langle\mathcal{X}, \mathcal{Y}\rangle$ to generate augmented data $\langle\mathcal{X}^*, \mathcal{Y}^*\rangle$. Then the labeled dataset is expanded as $\mathcal{L} = \mathcal{L} \cup \langle\mathcal{X}, \mathcal{Y}\rangle \cup \langle\mathcal{X}^*, \mathcal{Y}^*\rangle$. Formally, we define our task as: (1) construct a generator $\phi(\cdot)$ to implement sequence and label generation based on the actively sampled data $\mathcal{X}$ and its label $\mathcal{Y}$, (2) set a discriminator $d(\cdot)$ to yield the filtered generation, then (3) augment the labeled set as $\mathcal{L} = \mathcal{L} \cup \langle\mathcal{X}, \mathcal{Y}\rangle \cup d(\phi(\mathcal{X}, \mathcal{Y}))$.

2.2 Active Learning for Sequence Labeling

Active sequence labeling selects K most informative instances $\psi(\cdot, K, \gamma(\cdot))$ in each iteration, with the hope of maximally improving model performance with a fixed labeled budget. With the input sequence $\mathbf{x}$ of length T, we denote the model output as $f(\cdot|\mathbf{x};\theta)$. Our method is generic to any query policies $\gamma(\cdot)$. Below, we introduce several representative policies.

Least Confidence (LC) Culotta and McCallum (2005) measure the uncertainty of sequence models by the most likely predicted sequence. For a CRF model (Lafferty et al., 2001), we calculate γ with the predicted sequential label $\mathbf{y}^*$ as

$$\gamma^{\mathrm{LC}}(\mathbf{x}) = 1 - \max_{y^*}(P(\mathbf{y}^*|\mathbf{x};\theta), \quad (1)$$

where $\mathbf{y}^*$ is the Viterbi parse. For BERT (Devlin et al., 2019) with a token classification head, we adopt a variant of the least confidence measure:

$$\gamma^{\mathrm{LC'}}(\mathbf{x}) = \sum_{t=1}^{T}(1 - \max_{\mathbf{y}_t} P(\mathbf{y}_t|\mathbf{x};\theta)), \quad (2)$$

where $P(\mathbf{y}_t|\mathbf{x};\theta) = \mathrm{softmax}(f(\mathbf{y}_t|\mathbf{x};\theta))$.

Normalized Token Entropy (NTE) Another uncertainty measure for the query policy is normalized entropy (Settles and Craven, 2008), defined as:

$$\gamma^{\mathrm{TE}}(\mathbf{x}) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{m=1}^{M} P_m(\mathbf{y}_t|\mathbf{x},\theta)\log P_m(\mathbf{y}_t|\mathbf{x},\theta), \quad (3)$$

where $P_m(\mathbf{y}_t|\mathbf{x},\theta) = [\mathrm{softmax}(f(\mathbf{y}_t|\mathbf{x};\theta))]_m$.

Disagreement Sampling Query-by-committee (QBC) (Seung et al., 1992), is another approach for specifying the policy, where the unlabeled data can be sampled by the disagreement of the base models. The disagreement can be defined in several ways, here we take the vote entropy proposed by (Dagan and Engelson, 1995). Given a committee consist of C models, the vote entropy for input $\mathbf{x}$ is:

$$\gamma^{\mathrm{VE}}(\mathbf{x}) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{m=1}^{M}\frac{V_m(\mathbf{y}_t)}{C}\log\frac{V_m(\mathbf{y}_t)}{C}, \tag{4}$$

where $V_m(\mathbf{y}_t)$ is the number of models that predict the t-th token $\mathbf{x}_t$ as the label m.

3 The SeqMix Method

3.1 Overview

Given a corpus for sequence labeling, we assume the dataset contains a small labeled set $\mathcal{L}$ and a large unlabeled set $\mathcal{U}$ initially. We start from augmenting the seed set $\mathcal{L}$ with SeqMix. First, we adopt a pairing function $\zeta(\cdot)$ to find paired samples by traversing $\mathcal{L}$. Next, we generate mixed-labeled sequences via latent space linear interpolation with one of the approaches mentioned in Section 3.2. To ensure the semantic quality of the generated sequences, we use a discriminator $d(\cdot)$ to measure the perplexity of them and filter low-quality sequences out. Then we generate the extra labeled sequences $\mathcal{L}^* = \mathit{SeqMix}(\mathcal{L}, \alpha, \zeta(\cdot), d(\cdot))$ and get the augmented training set $\mathcal{L} = \mathcal{L} \cup \mathcal{L}^*$. The sequence labeling model θ is initialized on this augmented training set $\mathcal{L}$.

After that, the iterative active learning procedure begins. In each iteration, we actively select instances from $\mathcal{U}$ with a query policy $\gamma(\cdot)$ (Section 2.2) to obtain the top K samples $\mathcal{X} = \psi(\mathcal{U}, K, \gamma(\cdot))$. The newly selected samples will be labeled with $\mathcal{Y}$, and the batch of samples $\langle\mathcal{X}, \mathcal{Y}\rangle$ will be used for SeqMix. Again, we generate $\mathcal{L}^* = \mathit{SeqMix}(\langle\mathcal{X}, \mathcal{Y}\rangle, \alpha, \zeta(\cdot), d(\cdot))$ and expand the training set as $\mathcal{L} = \mathcal{L} \cup \mathcal{L}^*$. Then we train the model θ on the newly augmented set $\mathcal{L}$. The iterative active learning procedure terminates when a fixed number of iterations are reached. We summarize the above procedure in Algorithm 1.

Algorithm 1 The procedure of active sequence labeling augmentation via SeqMix

```
Input: Labeled seed set $\mathcal{L}$; Unlabeled set $\mathcal{U}$; Query function $\psi(\cdot, K, \gamma(\cdot))$; The sequence labeling model $\theta$; Beta distribution parameter $\alpha$; Pairing function $\zeta(\cdot)$; Discriminator function $d(\cdot)$.
// seed set augmentation
    $\mathcal{L}^* = \mathit{SeqMix}(\mathcal{L}, \alpha, \zeta(\cdot), d(\cdot))$
    $\mathcal{L} = \mathcal{L} \cup \mathcal{L}^*$
// model initialization
    $\theta$ = train $(\theta, \mathcal{L})$
// active learning iterations with augmentation
for round in active learning rounds do
    $\mathcal{X} = \psi(\mathcal{U}, K, \gamma(\cdot))$
    $\mathcal{U} = \mathcal{U} - \mathcal{X}$
    Annotate $\mathcal{X}$ to get $\langle\mathcal{X}, \mathcal{Y}\rangle$
    $\mathcal{L}^* = \mathit{SeqMix}(\langle\mathcal{X}, \mathcal{Y}\rangle, \alpha, \zeta(\cdot), d(\cdot))$
    $\mathcal{L} = \mathcal{L} \cup \langle\mathcal{X}, \mathcal{Y}\rangle \cup \mathcal{L}^*$
    $\theta$ = train $(\theta, \mathcal{L})$
end
Output: The sequence model trained with active data augmentation: $\theta$
```

3.2 Sequence Mixup in the Embedding Space

Mixup (Zhang et al., 2018) is a data augmentation method that implements linear interpolation in the input space. Given two input samples x_i, x_j along with the labels y_i, y_j, the mixing process is:

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j, \tag{5}$$

$$\tilde{y} = \lambda y_i + (1-\lambda)y_j, \tag{6}$$

where $\lambda \sim \mathit{Beta}(\alpha, \alpha)$ is the mixing coefficient. Through linear combinations on the input level of paired examples and their labels, Mixup regularizes the model to present linear behavior among the training data.

Mixup is not directly applicable to generate interpolated samples for text data, because the input space is discrete. To overcome this, SeqMix performs token-level interpolation in the embedding space and selects a token closest to the interpolated embedding. Specifically, SeqMix constructs a table of tokens $\mathcal{W}$ and their corresponding contextual embeddings $\mathcal{E}$[1]. Given two sequences $\mathbf{x_i} = \{\mathbf{w}_i^1, \cdots, \mathbf{w}_i^T\}$ and $\mathbf{x_j} = \{\mathbf{w}_j^1, \cdots, \mathbf{w}_j^T\}$ with their embedding representations $\mathbf{e}_{\mathbf{x_i}} = \{\mathbf{e}_i^1, \cdots, \mathbf{e}_i^T\}$ and $\mathbf{e}_{\mathbf{x_j}} = \{\mathbf{e}_j^1, \cdots, \mathbf{e}_j^T\}$, the t-th mixed token is the token whose embedding $\mathbf{e}^t$ is closest to the mixed embedding:

$$\mathbf{e}^t = \operatorname*{arg\,min}_{\mathbf{e}\in\mathcal{E}} \left\|\mathbf{e} - (\lambda\mathbf{e}_\mathbf{i}^\mathbf{t} + (1-\lambda)\mathbf{e}_\mathbf{j}^\mathbf{t})\right\|_2. \tag{7}$$

[1]The construction of $\{\mathcal{W}, \mathcal{E}\}$ are discussed in Appendix.

To get the corresponding $\mathbf{w}^t$, we can query the table $\{\mathcal{W}, \mathcal{E}\}$ using $\mathbf{e}^t$. The label generation is straightforward. For two label sequences $\mathbf{y}_i = \{\mathbf{y}_i^1, \cdots, \mathbf{y}_i^T\}$ and $\mathbf{y}_j = \{\mathbf{y}_j^1, \cdots, \mathbf{y}_j^T\}$, we get the t-th mixed label as:

$$\mathbf{y}^t = \lambda \mathbf{y}_i^t + (1-\lambda)\mathbf{y}_j^t, \tag{8}$$

where $\mathbf{y}_i^t$ and $\mathbf{y}_j^t$ are one-hot encoded labels.

Along with the above sequence mixup procedure, we also introduce a pairing strategy that selects sequences for mixup. The reason is that, in many sequence labeling tasks, the labels of interest are scarce. For example, in the NER and event detection tasks, the "O" label is dominant in the corpus, which do not refer to any entities or events of interest. We thus define the labels of interest as *valid labels*, *e.g.*, the non-"O" labels in NER and event detection, and design a sequence pairing function to select more informative parent sequences for mixup. Specifically, the sequence pairing function $\zeta(\cdot)$ is designed according to *valid label density*. For a sequence, its valid label density is defined as $\eta = \frac{n}{s}$, where n is the number of valid labels and s is the length of the sub-sequence. We set a threshold η_0 for $\zeta(\cdot)$, and the sequence will be considered as an eligible candidate for mixup only when $\eta \geq \eta_0$.

Based on the above token-level mixup procedure and the sequence pairing function, we propose three different strategies for generating interpolated labeled sequences. These strategies are shown in Figure 1 and described below:

Whole-sequence mixup As the name suggests, whole-sequence mixup (Figure 1(a)) performs sequence mixing at the whole-sequence level. Given two sequences $\langle \mathbf{x_i}, \mathbf{y_i} \rangle, \langle \mathbf{x_j}, \mathbf{y_j} \rangle \in \mathcal{L}$, they must share the same length without counting padding words. Besides, the paring function $\zeta(\cdot)$ requires that both the two sequences satisfy $\eta \geq \eta_0$. Then we perform mixup at all token positions, by employing Equation 7 to generate mixed tokens and Equation 8 to generate mixed labels (note that the mixed labels are soft labels).

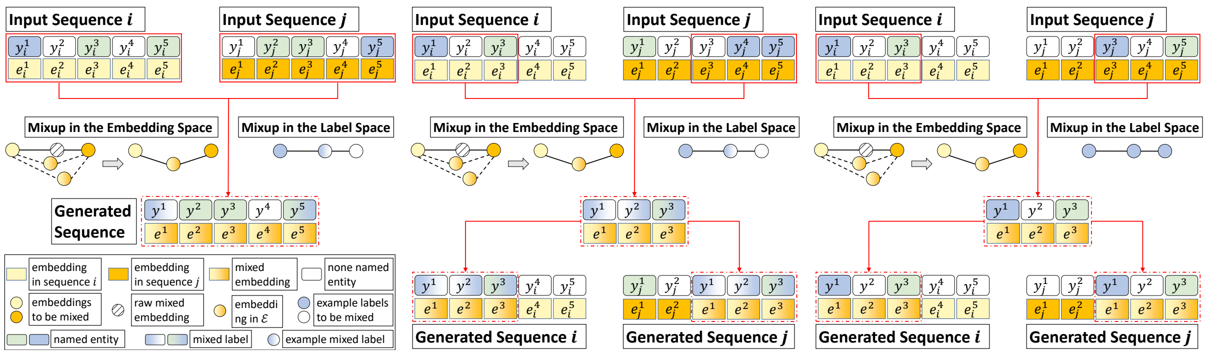
Sub-sequence mixup One drawback of the whole-sequence mixup is that it indiscriminately mixes over all tokens, which may include incompatible subsequences and generate implausible sequences. To tackle this, we consider sub-sequence mixup (Figure 1(b)) to mix sub-sequences of the parent sequences. It scans the original samples with a window of fixed-length s to look for

Algorithm 2 The generation procedure of SeqMix

```
Input: Labeled set $\mathcal{L} = \langle \mathcal{X}, \mathcal{Y} \rangle$; Beta distribution parameter $\alpha$; Pairing function $\zeta(\cdot)$; Discriminator function $d(\cdot)$; Number of expected generation $N$.
for $\langle \mathbf{x_i}, \mathbf{y_i} \rangle, \langle \mathbf{x_j}, \mathbf{y_j} \rangle, (i \neq j)$ in $\mathcal{L}$ do
    if $\zeta(\langle \mathbf{x_i}, \mathbf{y_i} \rangle, \langle \mathbf{x_j}, \mathbf{y_j} \rangle)$ then
        $\lambda \sim Beta(\alpha, \alpha)$
        // mixup the target sub-sequences
        for $t = 1, \cdots, T$ do
            Calculate $\mathbf{e}^t$ by Eq. (7);
            Get corresponding token $\mathbf{w}^t$ for $\mathbf{e}^t$;
            Calculate $\mathbf{y}^t$ by Eq. (8).
        end
        $\tilde{\mathbf{x}}_{sub} = \{\mathbf{w}^1, \cdots, \mathbf{w}^T\}$
        $\tilde{\mathbf{y}}_{sub} = \{\mathbf{y}^1, \cdots, \mathbf{y}^T\}$
        // replace the original sequences
        for $k$ in $\{i, j\}$ do
            $\tilde{\mathbf{x}}_k = \mathbf{x}_k - \mathbf{x}_{\mathbf{k}sub} + \tilde{\mathbf{x}}_{sub}$
            $\tilde{\mathbf{y}}_k = \mathbf{y}_k - \mathbf{y}_{\mathbf{k}sub} + \tilde{\mathbf{y}}_{sub}$
            if $d(\tilde{\mathbf{x}_\mathbf{k}})$ then
                $\mathcal{L}^* = \mathcal{L}^* \cup \langle \tilde{\mathbf{x}}_k, \tilde{\mathbf{y}}_k \rangle$
            end
            if $|\mathcal{L}^*| \geq N$ then
                break
            end
        end
    end
end
Output: Generated sequences and labels $\mathcal{L}^*$
```

paired sub-sequences. Denote the sub-sequences of $\langle \mathbf{x_i}, \mathbf{y_i} \rangle, \langle \mathbf{x_j}, \mathbf{y_j} \rangle$ as $\mathbf{X}_{\mathbf{i}sub} = \{\mathbf{x}_{\mathbf{i}sub}^1, \ldots, \mathbf{x}_{\mathbf{i}sub}^s\}$, $\mathbf{X}_{\mathbf{j}sub} = \left\{\mathbf{x}_{\mathbf{j}sub}^1, \ldots, \mathbf{x}_{\mathbf{j}sub}^s\right\}$. If $\exists\ \mathbf{x}_{\mathbf{i}sub} \in \mathbf{X}_{\mathbf{i}sub}$, $\mathbf{x}_{\mathbf{j}sub} \in \mathbf{X}_{\mathbf{j}sub}$, such that their $\eta \geq \eta_0$, we have $\zeta(\langle \mathbf{x_i}, \mathbf{y_i} \rangle, \langle \mathbf{x_j}, \mathbf{y_j} \rangle) =$ True. Then the sub-sequences $\mathbf{x}_{\mathbf{i}sub}$ and $\mathbf{x}_{\mathbf{j}sub}$ are mixed as Figure 1(b). The mixed sub-sequence and labels will replace the original parts of the parent samples, and the other parts of the parent samples remain unchanged. In this way, sub-sequence mixup is expected to keep the syntax structure of the original sequence, while providing data diversity.

Label-constrained sub-sequence mixup can be considered as a special case of sub-sequence mixup, where the constraints inherit sub-sequence mixup, and further require that the sub-sequence labels are consistent. As Figure 1(c) shows, after mixing such paired samples, the generation will just update the tokens of the sub-sequences while keeping the labels the same as before. Hence, this



(a) Whole sequence mixup (b) Sub-sequence mixup (c) Label-constrained sub-sequence mixup

Figure 1: Illustration of the three variants of SeqMix. We use $s = 5, \eta_0 = \frac{3}{5}$ for whole-sequence mixup and $s = 3, \eta_0 = \frac{2}{3}$ for sub-sequence mixup and label-constrained sub-sequence mixup. The solid red frames indicate paired sequences or sub-sequences, and the red dotted frames indicate generated sequence or sub-sequence. In the original sequences, the parts not included in the solid red frames will be unchanged in the generated sequences. For the mixup in the embedding space, we take the embedding in $\mathcal{E}$ which is closest to the raw mixed embedding as the generated embedding. For the mixup in the label space, the mixed label can be used as the pseudo label.

version is called label-constrained sub-sequence mixup.

Comparing the three variants, label-constrained sub-sequence mixup gives the most restrictions to pairing parent samples, sub-sequence mixup sets the sub-sequence-level pattern, while whole-sequence mixup just requires $\eta \geq \eta_0$ for the sequences with the same length.

3.3 Scoring and Selecting Plausible Sequences

During sequence mixup, the mixing coefficient λ determines the strength of interpolation. When λ approximates 0 or 1, the generated sequence will be similar to one of the parent sequences, while the λ around 0.5 produces relatively diverse generation. However, generating diverse sequences means low-quality sequences can be generated, which can provide noisy contextual information and hurt model performance.

To maintain the quality of mixed sequences, we set a discriminator to score the perplexity of the sequences. The final generated sequences will consist of only the sequences that pass the sequence quality screening. For the screening, we utilize a language model GPT-2 (Radford et al., 2019) to score sequence $\mathbf{x}$ by computing its perplexity:

$$\text{Perplexity}(\mathbf{x}) = 2^{-\frac{1}{T}\sum_{i=1}^{T}\log p(w_i)}, \qquad (9)$$

where T is the number of tokens before padding, w_i is the i-th token of sequence $\mathbf{x}$. Based on the perplexity and a score range $[s_1, s_2]$, the discriminator can give judgment for sequence $\mathbf{x}$:

$$d(\mathbf{x}) = \mathbb{1}\left\{s_1 \leq \text{Perplexity}\left(\mathbf{x}\right) \leq s_2\right\}. \qquad (10)$$

The lower the perplexity score, the more natural the sequence. However, the discriminator should also consider the regularization effectiveness and the generation capacity. Hence, a blind low perplexity setting is undesirable. The overall sequence mixup and selection procedure is illustrated in Algorithm 2.

4 Experiments

4.1 Experiment Setup

Datasets. We conduct experiments on three sequence labeling datasets for the named entity recognition (NER) and event detection tasks.
(1) **CoNLL-03** (Tjong Kim Sang and De Meulder, 2003) is a corpus for NER task. It provides four named entity types: persons, locations, organizations, and miscellaneous.[2]
(2) **ACE05** is a corpus for event detection. It provides 8 event types and 33 subtypes. We study the event trigger detection problem, which aims to identify trigger tokens in a sentence.
(3) **Webpage** (Ratinov and Roth, 2009) is a NER corpus with 20 webpages related to computer science conference and academic websites. It inherits the entity types from CoNLL-03.
Data Split. To investigate low-resource sequence labeling, we randomly take 700 labeled sentences

[2]We take the English version as our target corpus.

from the original CoNLL-03 dataset as the training set. For ACE05 and WebPage dataset, the annotation is sparse, so we conduct experiments on their original dataset without further slicing.

We set 6 data usage percentiles for the training set in each corpus. The sequence model is initialed on a small seed set, then it performs five iterates of active learning. For the query policy, we use random sampling and the three active learning policies mentioned in Section 2.2. The machine learning performance is evaluated by F_1 score for each data usage percentile.

Parameters. We use BERT-base-cased for the NER task as the underlying model, and BERT-base-multilingual-cased for the event trigger detection task. We set the max length as 128 to pad the varying-length sequences. The learning rate of the underlying model is 5e-5, and the batch size is 32. We train them for 10 epochs at each data usage percentile. For the parameters of SeqMix, we set $\alpha = 8$ to sample λ from $Beta(\alpha, \alpha)$. We use the sub-sequence window length $s = \{5, 5, 4\}$, the valid label density $\eta_0 = \{0.6, 0.2, 0.5\}$ for CoNLL-03, ACE05 and Webpage, respectively. The augment rate is set as 0.2, and the discriminator score range is set as $(0, 500)$. We also perform a detailed parameter study in Section 4.4.

4.2 Results

The main results are presented in Figure 2, where we use NTE sampling as the default active learning policy. From the result, it is clear that our method achieves the best performance consistently at each data usage percentile for all three datasets. The best SeqMix method (sub-sequence mixup with NTE sampling) outperforms the strongest active learning baselines by 2.95% on CoNLL-03, 2.27% on ACE05 and 3.75% on WebPage in terms of F_1 score on average. Moreover, the augmentation advantage is especially prominent for the seed set initialization stage where we only have a very limited number of labeled data. Through the augmentation, we improve the model performance from 68.65% to 80.71%, where the seed set is 200 labeled sequences and the augmentation provides extra 40 data points for CoNLL-03. The improvement is also significant on ACE05 (40.65% to 49.51%), and WebPage (55.18% to 71.67%), which indicates that our SeqMix can largely resolve the label scarcity issue in low-resource scenarios.

We also perform statistical significance tests for the above results. We use Wilcoxon Signed Rank Test (Wilcoxon, 1992), a non-parametric alternative to the paired t-test. This significance test fits our task as F-score is generally assumed to be not normally distributed (Dror et al., 2018), and non-parametric significance tests should be used in such a case. The results show that sub-sequence mixup and label-constrained sub-sequence mixup can provide a statistical significance (the confidence level $\alpha = 0.05$ and the number of data points $N = 6$) for all the comparisons with active learning baselines on used datasets. The whole-sequence mixup passes the statistical significance test with $\alpha = 0.1$ and $N = 6$ on CoNLL-03 and WebPage, but fails on ACE05.

Data Usage	200	300	400	500	600	700
(0, $+\infty$)	**81.15**	82.32	82.74	83.66	83.79	85.05
(0, 2000)	80.20	82.24	83.21	83.67	83.90	85.11
(0, 1000)	80.13	81.86	83.58	84.22	84.81	85.16
(0, 500)	80.71	**82.82**	**84.05**	**85.28**	**86.04**	**86.24**

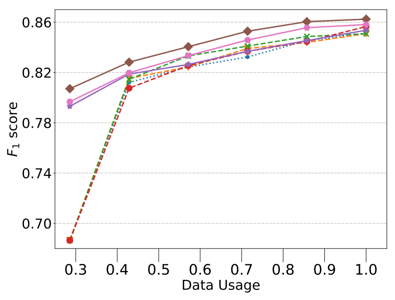
Table 1: The $F_1(\%)$ of sub-sequence mixup with NTE sampling in different discriminator score range, evaluated on CoNLL-03 with 700 data.

Among all the three SeqMix variants, sub-sequence mixup gives the overall best performance (label-constrained sub-sequence mixup achieves very close performance with sub-sequence mixup on ACE05 dataset), but whole-sequence mixup does not yield a consistent improvement to the original active learning method. This is because the whole-sequence mixup may generate semantically poor new sequences. Instead, the sub-sequence-level process reserves the original context information between the sub-sequence and the other parts of the whole sequence. Meanwhile, the updated sub-sequences inherit the original local informativeness, and introduce linguistic diversity to enhance the model's generalization ability.

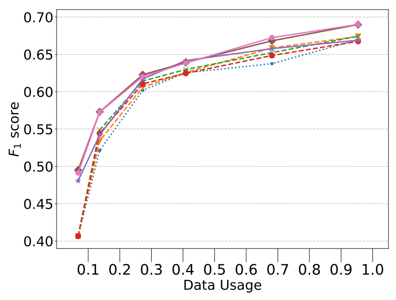
To justify that SeqMix can provide improvement to the active learning framework with various query policies, we employ different query policies with SeqMix augmentation under the same experiment setting as Figure 2(a). From Figure 3, we find that there is a consistent performance improvement when employing SeqMix with different query policies. As SeqMix achieves $\{2.46\%, 2.85\%, 2.94\%\}$ performance gain for random sampling, LC sampling and NTE sampling respectively.

4.3 Effect of Discriminator

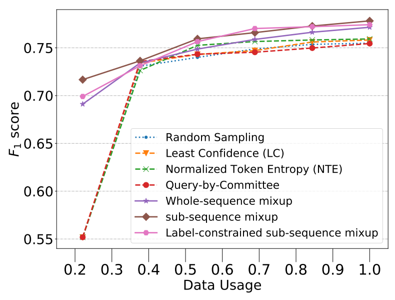
To verify the effectiveness of the discriminator, we conduct the ablation study on a subset of CoNLL-



(a) CoNLL-03 (700 labeled data)



(b) ACE05 (14k labeled data)



(c) WebPage (385 labeled data)

Figure 2: The F_1 score of test set in terms of data usage on CoNLL-03, ACE05 and WebPage.

Data Usage	200	300	400	500	600	700	Average
$r = 0.2$	**80.22 (+0.76)**	82.23(+0.43)	**83.61 (+0.61)**	84.62 (+0.53)	85.16 (+0.10)	85.22 (-0.11)	**+ 0.39**
$r = 0.4$	79.71 (+0.25)	**82.48(+0.68)**	82.66 (-0.34)	83.46 (-0.63)	84.79 (-0.27)	85.24 (-0.09)	- 0.07
$r = 0.6$	79.40 (-0.06)	82.07(+0.27)	83.34 (+0.34)	**84.75 (+0.66)**	**85.43 (+0.37)**	**85.50 (+0.17)**	+ 0.29
$r = 0.8$	79.48 (+0.02)	81.63(-0.17)	82.80 (-0.20)	83.29 (-0.80)	84.54 (-0.52)	85.32 (-0.01)	- 0.28
$r = 1.0$	78.51 (-0.95)	80.58(-1.22)	82.59 (-0.41)	84.31 (+0.22)	85.36 (+0.30)	85.37 (+0.04)	- 0.34

Table 2: The F_1 score with variant augment rate r. The value in the parentheses is the difference with the average F_1 for corresponding data usage. The last column presents the average F_1 difference for each learning rate r.

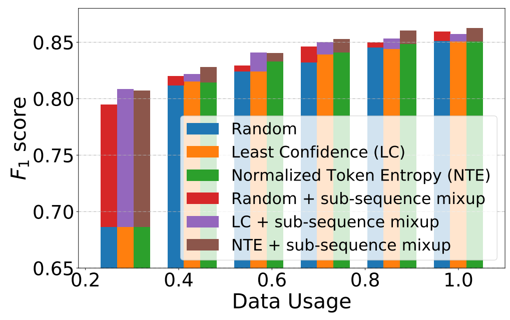
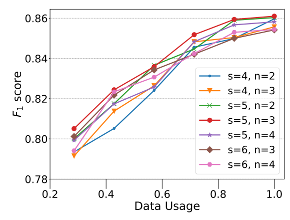


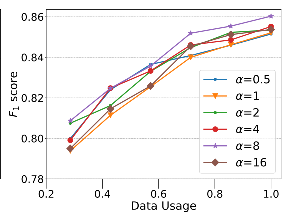
Figure 3: The improvements to various active learning approaches provided by SeqMix.

03 with 700 labeled sequences. We use sub-sequence mixup with NTE sampling as the backbone and change the perplexity score range of the discriminator. We start from the seed set with 200 labeled data, then actively query 100 data in each learning round and repeat 5 rounds in total.

The result in Table 1 demonstrates the discriminator provides a stable improvement for the last four data usage percentiles, and the discriminator with score range $(0, 500)$ can boost the model by 1.07% F_1 score, averaged by all the data usage percentiles. The comparison between 3 different score thresholds demonstrates the lower the perplexity, the better the generation quality. As a result, the final F_1 score becomes higher with the better generated tokens. Actually, we can further narrow down the score range to get more performance improvement in return, but the too strict constraints will slow down the generation in practice and reduce the number of generated samples.



(a) F_1 with several combination of s and n



(b) F_1 with different α

Figure 4: Parameter Search for SeqMix

4.4 Parameter Study

In this subsection, we study the effect of several key parameters.

Augment rate r. We vary the augment rate $r = \frac{|\mathcal{L}^*|}{|\psi(\mathcal{U},K,\gamma(\cdot))|}$ in $\{0.2, 0.4, 0.6, 0.8, 1.0\}$ and keep the number of initial data usage same to investigate the effect of augment rate for data augmentation. Table 2 shows that $r \leq 0.6$ can provide better F_1 improvement. The model with $r = 0.2$ surpasses the model with $r = 1.0$ by 0.73%, evaluated by the average F_1 score for all the data usage percentiles. This result indicates that the model appreciates moderate augmentation more. However, the performance variance based on the augment rate is not prominent compared to the improvement provided by SeqMix to the active learning framework.

Valid tag density η_0. We search the valid tag density η_0 as Section 3.2 defined by varying the sub-sequence window length s and the required number of valid tag n within the window. The results in Figure 4(a) illustrate the combination $(s = 5, n = 3)$ outperforms other settings. When s is too small, the window usually truncates the continuous clause, thus cutting off the local syntax or semantic information. When s is too large, sub-sequence mixup tends to behave like whole-sequence mixup, where the too long sub-sequence generation can hardly maintain the rationality of syntax and semantics as before. The high η_0 with long window length may result in an insufficient amount of eligible parent sequences. Actually, even with a moderate augment rate $\alpha = 0.2$, the combination $(s = 6, n = 5)$ has been unable to provide enough generation.

Mixing parameter α. We show the performance with different α in Figure 4(b). The parameter α decides the distribution $\lambda \sim Beta(\alpha, \alpha)$, and the coefficient λ directly involved the mixing of tokens and labels. Among the values $\{0.5, 1, 2, 4, 8, 16\}$, we observed $\alpha = 8$ presents the best performance. It outperforms the second-best parameter setting 0.49% by average. From the perspective of Beta distribution, larger α will make the sampled λ more concentrated around 0.5, which assigns more balance weights to the parent samples to be mixed. In this way, the interpolation produces encoded token with further distance to both the parent samples, thus introduces a more diverse generation.

4.5 Case Study

Figure 5 presents a generation example via sub-sequence mixup. For the convenience of presentation, we set the length of sub-sequence $s = 3$ and the valid label density threshold $\eta_0 = \frac{2}{3}$. The two input sequences get paired for their eligible sub-sequences "`COLORADO 10 St`" and "`Slovenia , Kwasniewski`". The sub-sequences are mixed by $\lambda = 0.39$ in this case, which is sampled from $Beta(\alpha, \alpha)$. Then the generated sub-sequence "`Ohio ( novelist`" replaces the original parts in the two input sequences. Among the generated tokens, "`Ohio`" inherits the label `B-ORG` from "`COLORADO`" and the label `B-LOC` from "`Slovenia`", and the distribution $Beta(\alpha, \alpha)$ assigns the two labels with weights $\lambda = 0.39$ and $(1 - \lambda) = 0.61$. The open parenthesis is produced by the mixing of a digit and a punctuation mark, and keeps the label `O` shared by its parents. Similarly, the token "`novelist`" generated by "`St`" and "`Kwasniewski`" gets a mixed label from `B-ORG` and `B-PER`.

The discriminator then evaluates the two generated sequences. The generated sequence i is not reasonable enough intuitively, and its perplexity score 877 exceeds the threshold, so it is not added into the training set. The generated sequence j retains the original syntax and semantic structure much better. Although the open parenthesis seems strange, it plays a role as the comma in the original sequence to separate two clauses. This generation behaves closely to a normal sequence and earns 332 perplexity score, which permits its incorporation into the training set.

5 Related Work

Active Sequence Labeling Sequence labeling has been studied extensively for different NLP problems. Different neural architectures has been proposed (Huang et al., 2015; Lample et al., 2016; Peters et al., 2018; Akbik et al., 2018) in recent years, which have achieved state-of-the-art performance in a number of sequence labeling tasks. However, these neural models usually require exhaustive human efforts for generating labels for each token, and may not perform well in low-resource settings. To improve the performance of low-resource sequence labeling, several approaches have been applied including using semi-supervised methods (Clark et al., 2018; Chen et al., 2020b), external weak supervision (Lison et al., 2020; Liang et al., 2020; Ren et al., 2020; Zhang et al., 2019; Yu et al., 2020) and active learning (Shen et al., 2017; Hazra et al., 2019; Liu et al., 2018; Fang et al., 2017; Gao et al., 2019). In this study, we mainly focus on active learning approaches which select samples based on the query policy design. So far, various uncertainty-based (Scheffer et al., 2001; Culotta and McCallum, 2005; Kim et al., 2006) and committee-based approaches (Dagan and Engelson, 1995) have been proposed for improving the sample efficiency. More recently, Shen et al. (2017); Hazra et al. (2019); Liu et al. (2018); Fang et al. (2017) further improve the aforementioned active learning approaches to improve the sampling diversity as well as the generalization ability of models on low-resource scenarios. These works mainly claim the sample efficiency provided by the active learning approach but do not study data augmentation for active sequence labeling.

Interpolation-based Regularizations Mixup implements interpolation in the input space to regularize models (Zhang et al., 2018). Recently,

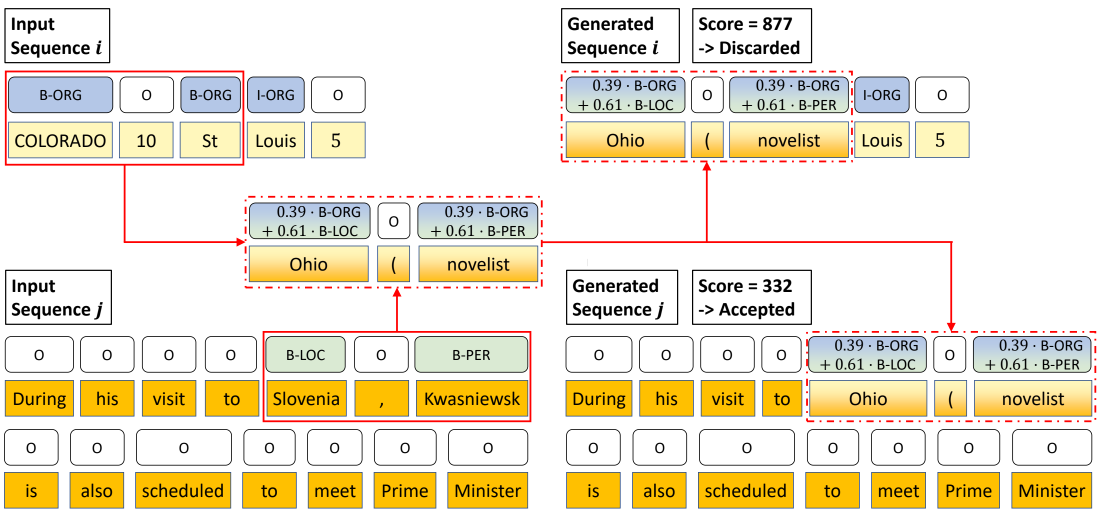


Figure 5: A generation case of sub-sequence mixup.

the Mixup variants (Verma et al., 2019; Summers and Dinneen, 2019; Guo et al., 2019b) turn to perform interpolation in the hidden space to capture higher-level information. Guo et al. (2019a); Chen et al. (2020a) apply hidden-space Mixup for text classification. These works, however, have not explored how to perform mixup for sequences with token-level labels, nor do they consider the quality of the mixed-up samples.

Text Augmentation Our work is also related to text data augmentation. Zhang et al. (2015); Wei and Zou (2019) utilize heuristic approaches including synonym replancement, random insertion, swap and deletion for text augmentation, Kafle et al. (2017); Silfverberg et al. (2017) employ heuristic rules based on specific task, Hu et al. (2017) propose to augment text data in an encoder-decoder manner. Very recently, (Anaby-Tavor et al., 2020; Kobayashi, 2018) harness the power of pre-trained language models and augmenting the text data based on contextual patterns. Although these methods can augment the training set and improve the performance of text classification model, they fail to *generate sequences and labels simultaneously*, thus cannot be adapted to our problem where token-level labels are required during training. Instead, in our study, we propose a new framework SeqMix for data augmentation to facilitate sequence labeling task. Our method can generate token-level labels and preserve the semantic information in the augmented sentences. Moreover, it can be naturally combined with existing active learning approaches and further promote the performance.

6 Conclusion

We propose a simple data augmentation method SeqMix to enhance active sequence labeling. By performing sequence mixup in the latent space, SeqMix improves data diversity during active learning, while being able to generate plausible augmented sequences. This method is generic to different active learning policies and various sequence labeling tasks. Our experiments demonstrate that SeqMix can improve active learning baselines consistently for NER and event detection tasks; and its benefits are especially prominent in low-data regimes. For future research, it is interesting to enhance SeqMix with language models during the mixup process, and harness external knowledge for further improving diversity and plausibility.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

Ateret Anaby-Tavor, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. 2020. Do not have enough data? deep learning to the rescue! In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 7383–7390.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21.

Jiaao Chen, Zichao Yang, and Diyi Yang. 2020a. MixText: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2147–2157.

Luoxin Chen, Weitong Ruan, Xinyue Liu, and Jianhua Lu. 2020b. SeqVAT: Virtual adversarial training for semi-supervised sequence labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8801–8811.

Eunah Cho, He Xie, and William M. Campbell. 2019. Paraphrase generation for semi-supervised learning in NLU. In *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 45–54.

Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. 2018. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925.

Aron Culotta and Andrew McCallum. 2005. Reducing labeling effort for structured prediction tasks. In *Proceedings of the 20th AAAI conference on Artificial intelligence*, volume 5, pages 746–751.

Ido Dagan and Sean P Engelson. 1995. Committee-based sampling for training probabilistic classifiers. In *Machine Learning Proceedings 1995*, pages 150–157.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392.

Meng Fang, Yuan Li, and Trevor Cohn. 2017. Learning how to active learn: A deep reinforcement learning approach. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 595–605.

Ning Gao, Nikos Karampatziakis, Rahul Potharaju, and Silviu Cucerzan. 2019. Active entity recognition in low resource settings. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, page 2261–2264.

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019a. Augmenting data with mixup for sentence classification: An empirical study. *arXiv preprint arXiv:1905.08941*.

Hongyu Guo, Yongyi Mao, and Richong Zhang. 2019b. Mixup as locally linear out-of-manifold regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3714–3722.

Rishi Hazra, Shubham Gupta, and Ambedkar Dukkipati. 2019. Active^2 learning: Actively reducing redundancies in active learning methods for sequence tagging. *arXiv preprint arXiv:1911.00234*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Kushal Kafle, Mohammed Yousefhussien, and Christopher Kanan. 2017. Data augmentation for visual question answering. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 198–202.

Seokhwan Kim, Yu Song, Kyungduk Kim, Jeong-Won Cha, and Gary Geunbae Lee. 2006. MMR-based active machine learning for bio named entity recognition. In *Proceedings of the Human Language Technology Conference of the NAACL,*, pages 69–72.

Sosuke Kobayashi. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457.

John D Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

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.

Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. 2020. Bond: Bert-assisted open-domain named entity recognition with distant supervision. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 1054–1064.

Pierre Lison, Jeremy Barnes, Aliaksandr Hubin, and Samia Touileb. 2020. Named entity recognition without labelled data: A weak supervision approach. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1518–1533.

Ming Liu, Wray Buntine, and Gholamreza Haffari. 2018. Learning how to actively learn: A deep imitation learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1874–1883.

Lluís Màrquez, Pere Comas, Jesús Giménez, and Neus Català. 2005. Semantic role labeling as sequential tagging. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 193–196.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. 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.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155.

Wendi Ren, Yinghao Li, Hanting Su, David Kartchner, Cassie Mitchell, and Chao Zhang. 2020. Denoising multi-source weak supervision for neural text classification. In *Findings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

Tobias Scheffer, Christian Decomain, and Stefan Wrobel. 2001. Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, pages 309–318. Springer.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079.

H. S. Seung, M. Opper, and H. Sompolinsky. 1992. Query by committee. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, page 287–294.

Yanyao Shen, Hyokun Yun, Zachary Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017. Deep active learning for named entity recognition. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 252–256.

Miikka Silfverberg, Adam Wiemerslage, Ling Liu, and Lingshuang Jack Mao. 2017. Data augmentation for morphological reinflection. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 90–99.

Cecilia Summers and Michael J Dinneen. 2019. Improved mixed-example data augmentation. In *2019 IEEE Winter Conference on Applications of Computer Vision*, pages 1262–1270.

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 at HLT-NAACL 2003*, pages 142–147.

Katrin Tomanek, Joachim Wermter, and Udo Hahn. 2007. An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 486–495.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. 2019. Manifold mixup: Better representations by interpolating hidden states. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6438–6447. PMLR.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388.

Frank Wilcoxon. 1992. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. 2019. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*.

Sen Yang, Dawei Feng, Linbo Qiao, Zhigang Kan, and Dongsheng Li. 2019. Exploring pre-trained language models for event extraction and generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5284–5294.

Yue Yu, Yinghao Li, Jiaming Shen, Hao Feng, Jimeng Sun, and Chao Zhang. 2020. Steam: Self-supervised taxonomy expansion with mini-paths. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 1026–1035.

Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

Shanshan Zhang, Lihong He, Eduard Dragut, and Slobodan Vucetic. 2019. How to invest my time: Lessons from human-in-the-loop entity extraction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 2305–2313.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Yizhe Zhang, Zhe Gan, and Lawrence Carin. 2016. Generating text via adversarial training. In *NIPS workshop on Adversarial Training*.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 647–657.

A Information for Dataset

A.1 Dataset Collection

Here we list the link to datasets used in our experiments.

- **CoNLL-03**: https://github.com/synalp/NER/tree/master/corpus/CoNLL-2003.
- **ACE05**: We are unable to provide the downloadable version due to it is not public. This corpus can be applied through the website of LDC: https://www.ldc.upenn.edu/collaborations/past-projects/ace.
- **Webpage**: Please refer the link in the paper (Ratinov and Roth, 2009).

A.2 Dataset Split

All the mentioned dataset has been split into **train/validate/test** set in the released version. We keep consistent with the validation set and the test set in our experiment. For the active learning paradigm, we split the training set as Table 3. The active learners are initialized on the seed set, then they implement 5 active learning rounds.

B Baseline Settings

For the baselines, we take random sampling and 3 active learning approaches – LC sampling, NTE sampling, and QBC sampling as Section 2.2.

C Implementation Details of SeqMix

We implement bert-base-cased as the underlying model for the NER task and bert-base-multilingual-cased as the underlying model for the event detection task. We use the model from Huggingface Transformer codebase[3], and the repository[4] to fine-tune our model for sequence labeling task.

C.1 Number of Parameters

In our model, we use **bert-base-cased** and **bert-base-multilingual-cased** both of them occupy 12-layer, 768-hidden, 12-heads with 110M parameters.

[3]https://github.com/huggingface/transformers

[4]https://github.com/kamalkraj/BERT-NER

C.2 Adapting BERT for sequence labeling task

To fine-tune on sequence labeling tasks, a dropout layer ($p = 0.1$) and a linear (token-level) classification layer is built upon the pre-trained model.

C.3 SeqMix Details

In Section 3.2, we construct a table of tokens $\mathcal{W}$ and their corresponding contextual embedding $\mathcal{E}$. For our underlying BERT model, we use the vocabulary provided by the tokenizer to build up $\mathcal{W}$, and the embedding initialized on the training set as $\mathcal{E}$.

We also need to construct a special token collection to exclude some generation in the process of sequence mixing. For example, BERT places token `[CLS]` and `[SEP]` at the starting position and the ending position for sentence, and pad the inputs with `[PAD]`. We exclude these disturbing tokens and the parent tokens.

C.4 Parameter Settings

The key parameters setting in our framework are stated here: (1) The number of active learning round is 5 for all the three datasets, but the size of seed set and the number of samples in each round differs from the dataset. We list the specific numbers as Table 3. (2) The sub-sequence window length s and the valid label density threshold η_0 vary from the datasets. For CoNLL-03, $s = 5$, $\eta_0 = 0.6$; for ACE05, $s = 5$, $\eta_0 = 0.2$; for WebPage, $s = 4$, $\eta_0 = 0.5$. (3) We set $\alpha = 8$ for the *Beta* distribution. (4) The discriminator score range is set as $(0, 500)$ for all the datasets. (5) For BERT configuration, we choose 5e-5 for learning rate, 128 for padding length, 32 for batch size, 0.1 for dropout rate, 1e-8 for ϵ in Adam. At each data usage point, we train the model for 10 Epochs. (6) We set $\mathcal{C} = 3$ for the QBC query policy.

D Details of Experiments

We take following criteria to evaluate the sequence labeling task. A named entity is correct only if it is an exact match of the corresponding entity in the data file. An event trigger is correct only if the span and type match with golden labels. Based on the above metric, we evaluate F_1 score in our experiments.

D.1 Performance on Development Set

Table 4 to Table 6 shows the model performance on the validation set. The data usage in these tables

Dataset	# of Entity Types	# of Seed Set	Sampling Rounds	# of Each Round Samples	# of Dev	# of Test
CoNLL-03	4	200	5	100	3250	3453
ACE05	29	1k	5	{1k, 2k, 2k, 4k, 4k}	873	711
Webpage	4	85	5	60	99	135

Table 3: The information for benchmarks in our experiments.

Data Usage	200	300	400	500	600	700
Random Sampling	69.03	83.28	84.93	85.50	85.79	86.62
LC Sampling	69.03	83.78	84.55	85.88	86.04	86.73
NTE Sampling	69.03	83.60	85.00	85.47	86.19	86.83
QBC Sampling	69.03	83.33	84.52	85.30	86.27	86.60
Sub-sequence mixup	81.69	85.28	85.95	86.52	87.07	87.44

Table 4: Validation F_1 of CoNLL-03

Data Usage	85	145	205	265	325	385
Random Sampling	0	27.52	34.41	34.83	37.93	35.73
LC Sampling	0	28.84	32.88	34.22	38.78	38.11
NTE Sampling	0	22.44	34.81	33.74	36.59	38.27
QBC Sampling	0	23.88	32.18	34.17	36.56	35.66
Sub-sequence mixup	14.35	33.74	34.70	36.22	39.74	38.25

Table 6: Validation F_1 of WebPage

Data Usage	1000	2000	4000	6000	10000	14000
Random Sampling	48.16	59.10	63.13	64.95	66.23	67.12
LC Sampling	48.16	59.33	63.22	65.04	66.24	66.92
NTE Sampling	48.16	59.72	63.17	65.53	66.78	67.24
QBC Sampling	48.16	59.01	62.79	64.89	66.20	66.91
Sub-sequence mixup	56.51	61.62	63.65	65.83	67.54	67.98

Table 5: Validation F_1 of ACE05

refers to the number of labeled data, excluding the augmentation data. Sub-sequence mixup is trained with $(1+\alpha)$ times data, where the α denotes the augment rate. Note that WebPage is a very limited dataset, there is a big difference between the performance on the validation set and the test set. We average each experiment by 5 times.

D.2 Computing Infrastructure

We implement our system on *Ubuntu 18.04.3 LTS* system. We run our experiments on an Intel(R) Xeon(R) CPU @ 2.30GHz and NVIDIA Tesla P100-PCIe with 16 GB HBM2 memory. The NVIDIA-SMI version is 418.67 and the CUDA version is 10.1.

D.3 Average Runtime

For the 5-round active learning with SeqMix augmentation, our program runs about 500 seconds for WebPage dataset, 1700 seconds for the CoNLL slicing dataset, and 3.5 hours for ACE 2005. If the QBC query policy used, all the runtime will be multiplied about 3 times.

D.4 Hyper parameter Search

For the discriminator score range, we first examine the perplexity score distribution of the CoNLL training set. Then determine an approximate score range $(0, 2000)$ first. We linearly split score ranges below 2000 to conduct parameter study and report the representative ranges in Section 4.3. Given the consideration to the generation speed and the augment rate setting, we finally choose 500 as the upper limit rather than a too narrow score range setting.

For the mixing coefficient λ, we follow (Zhang et al., 2018) to sample it from *Beta*(α, α) and explore α ranging from $[0.5, 16]$. We present this parameter study in Section 4.4. The result shows different α did not influence the augmentation performance much.

For the augment rate and the valid tag density, we also have introduced the parameter study in Section 4.4.