Local Additivity Based Data Augmentation for Semi-supervised NER

Jiaao Chen*, **Zhenghui Wang***, **Ran Tian**[1], **Zichao Yang**[2], **Diyi Yang**
Georgia Institute of Technology, [1]ASIT Japan / Google, [2]Citadel Securities
{jchen896,zhwang,dyang888}@gatech.edu

Data augmentation

- (1) adversarial attacks at token-levels such as word substitutions or adding noise
  - struggles to create diverse augmented samples with very few word replacements
- (2) paraphrasing at sentence-levels such as back translations or submodular optimized models
  - fails to maintain the labels at the token-level in those paraphrased sentences（widely utilized in many NLP tasks like text classification）

Mixup

Zhang et al. (2018) proposed a data augmentation technique called *mixup*, which trained an image classifier on linear interpolations of randomly sampled image data. Given a pair of data points $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}', \mathbf{y}')$, where $\mathbf{x}$ denotes an image in raw pixel space, and $\mathbf{y}$ is the label in a one-hot representation, *mixup* creates a new sample by interpolating images and their corresponding labels:

$$\tilde{\mathbf{x}} = \lambda \mathbf{x} + (1 - \lambda)\mathbf{x}',$$
$$\tilde{\mathbf{y}} = \lambda \mathbf{y} + (1 - \lambda)\mathbf{y}',$$

where λ is drawn from a Beta distribution. *mixup* trains the neural network for image classification by minimizing the loss on the virtual examples. In experiments, the pairs of images data points $(\mathbf{x}, \mathbf{y})$ and $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ are *randomly* sampled. By assuming all the images are mapped to a low dimension manifold through a neural network, linearly interpolating them creates a virtual vicinity distribution around the original data space, thus improving the generalization performance of the classifier trained on the interpolated samples.

As we found in empirical experiments, it is challenging to directly apply such mixup technique to sequence labeling, and improper interpolations may mislead the model.

For instance, random sampling in mixup may inject too much noise by interpolating data points far away from each other, hence making it fail on sequence labeling.

LADA

For a given sentence with n tokens $\mathbf{x} = \{x_1, ..., x_n\}$, denote the corresponding sequence label as $\mathbf{y} = \{y_1, ..., y_n\}$. In this paper, we use NER as the working example to introduce our model, in which the labels are the entities types. We randomly sample a pair of sentences from the corpus, $(\mathbf{x}, \mathbf{y})$ and $(\mathbf{x}', \mathbf{y}')$, and then compute the interpolations in the hidden space using a L-layer encoder $\mathbf{F}(.; \theta)$. The hidden representations of $\mathbf{x}$ and $\mathbf{x}'$ up to the m-th layer are given by:

$$\mathbf{h}^l = \mathbf{F}^l(\mathbf{h}^{l-1}; \theta), l \in [1, m],$$
$$\mathbf{h}'^l = \mathbf{F}^l(\mathbf{h}'^{l-1}; \theta), l \in [1, m],$$

Here $\mathbf{h}^l = \{h_1, ..., h_n\}$ refer to the hidden representations at the l-th layer and is the concatenation of token representations at all positions. We use $\mathbf{h}^0, \mathbf{h}'^0$ to denote the word embedding of $\mathbf{x}$ and $\mathbf{x}'$ respectively. At the m-th layer, the hidden representations for each token in $\mathbf{x}$ are linearly interpolated with each token in $\mathbf{x}'$ by a ratio λ:

$$\tilde{\mathbf{h}}^m = \lambda \mathbf{h}^m + (1 - \lambda)\mathbf{h}'^m,$$

where the mixing parameter λ is sampled from a Beta distribution, i.e., $\lambda \sim \text{Beta}(\alpha, \alpha)$. Then $\tilde{\mathbf{h}}^m$ is fed to the upper layers:

$$\tilde{\mathbf{h}}^l = \mathbf{F}^l(\tilde{\mathbf{h}}^{l-1}; \theta), l \in [m + 1, L].$$

$\tilde{\mathbf{h}}^L$ can be treated as the hidden representations of a *virtual sample* $\tilde{\mathbf{x}}$, i.e., $\tilde{\mathbf{h}}^L = \mathbf{F}(\tilde{\mathbf{x}}; \theta)$.

In the meanwhile, their corresponding labels are linearly added with the same ratio:

$$\tilde{y}_i = \lambda y_i + (1 - \lambda) y'_i$$
$$\tilde{\mathbf{y}} = \{\tilde{y}_1, ..., \tilde{y}_n\}.$$

LADA

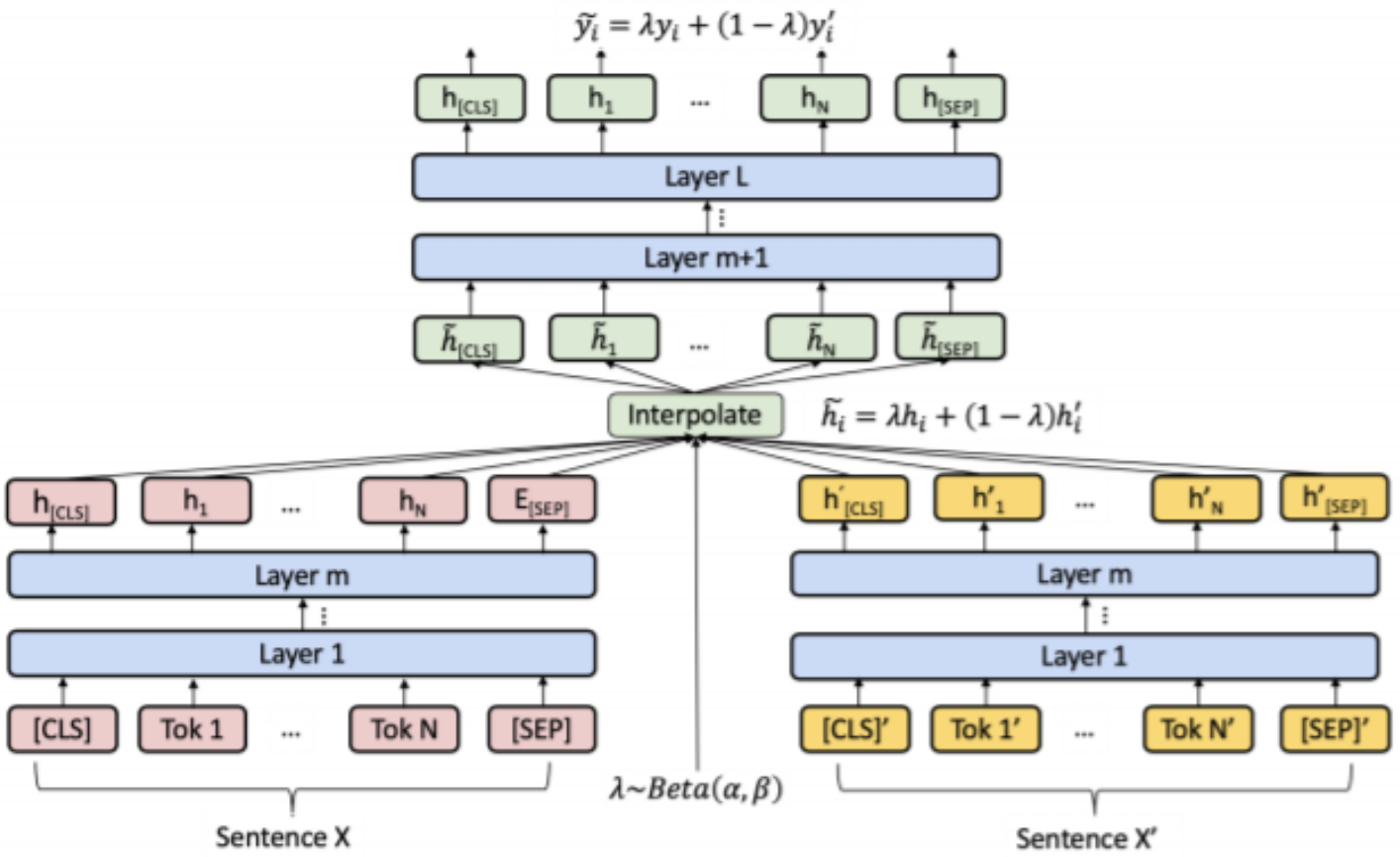

Figure 1: Overall Architecture of LADA. LADA takes in two sentences, linearly interpolates their hidden states h_i and h'_i at layer m with weight λ into $\tilde{h}_i$, and then continues forward passing to get encoded representations $\tilde{h}_i$, which are utilized in downstream tasks where the labels in each task are also mixed with weight λ.

Intra-LADA

- We hypothesize that mixing the sequence x with x' changes the context for all tokens and injects too much noise, hence making learning the labels for the tokens challenging.
- To reduce the noises from unrelated sentences, the most direct way is to construct x' using the same tokens from x but changing the orders and perform interpolations between them.

Let $\mathbf{Q} = \text{Permutations}((\mathbf{x}, \mathbf{y}))$ be the set including all possible permutations of $\mathbf{x}$, then

$$P_{\text{Intra}}(\mathbf{x}'|\mathbf{x}) = \frac{1}{n!}, \qquad (\mathbf{x}', \mathbf{y}') \in \mathbf{Q}. \quad (3)$$

In this case, each token x_i in $\mathbf{x}$ is actually interpolated with another token x_j in $\mathbf{x}$, while the context is unaltered. By sampling from P_{Intra}, we are essentially turning sequence level interpolation to token level interpolation, thus greatly reducing the complexity of the problem. From another perspective, Intra-LADA generates augmentations with different sentence structures using the same word set, which could potentially increase the model's robustness towards word orderings.

Intra-LADA restraints the context from changing, which could be limited in generating diverse augmented data. To overcome that, we propose Inter-LADA, where we sample a different sentence from the training set to perform interpolations.

Inter-LADA

Instead of interpolating within one sentence, Intra-LADA samples a different sentence $\mathbf{x}'$ from the training set to interpolate with $\mathbf{x}$. To achieve a trade-off between noise and regularization, we sample $\mathbf{x}'$ through a weighted combination of two strategies: k-nearest neighbors (kNNs) sampling and random sampling:

$$P_{\text{Inter}}(\mathbf{x}'|\mathbf{x}) = \begin{cases} \frac{\mu}{k}, & \mathbf{x}' \in \text{Neighbor}_k(\mathbf{x}), \\ \frac{1-\mu}{|S|}, & (\mathbf{x}', \mathbf{y}') \in \mathbf{S}, \end{cases} \tag{4}$$

where μ is the weight of combining two distributions. To get the kNNs, we use sentence-BERT (Reimers and Gurevych, 2019) to map each sentence $\mathbf{x}$ into a hidden space, then collect each sentence's kNNs using l^2 distance. For each sentence $\mathbf{x}$, we sample $\mathbf{x}'$ to mix up from the kNNs with probability μ and the whole training corpus with a probability $1-\mu$. When $\mathbf{x}'$ is sampled from the whole training corpus, it may be unrelated to $\mathbf{x}$, introducing large noise but also strong regularization on the model. When $\mathbf{x}'$ is sampled from the kNNs, $\mathbf{x}'$ shares similar, albeit different, context with $\mathbf{x}$, thus achieving good signal to noise ratio. By treating μ as a hyper-parameter, we can control the delicate trade-off between noise and diversity in regularizing the model.

In summary, Inter-LADA can improve both entity learning and context learning by interpolating more diverse data. Note that although we use NER as a working example , LADA can be applied to any sequence labeling models.

(1)As it shows, kNNs may contain the same entity words as the original sentence, but in different contexts

(2)Contexts from neighbor sentences can help detect the entities of the same type in a given sentence

(3)Neighbor sentences may contain the same words but in different forms

Sentence	Israel plays down fears of war with Syria.
Neighbours	Fears of an Israeli operation causes the redistribution of Syrian troops locations in Lebanon .
	Parliament Speaker Berri: Israel is preparing for war against Syria and Lebanon .
	Itamar Rabinovich , who as Israel's ambassador to Washington conducted unfruitful negotiations with Syria , told Israel Radio looked like Damascus wanted to talk rather than fight .

Table 1: *k*NNs of an example sentence. Entities in sentences are colored. Green means locations , red means persons , blue means organizations and yellow means miscellaneous.

Semi-supervised LADA

To further improve the performance of learning with less labeled data, we propose a novel LADA-based approach specifically for unlabeled data. Instead of looking for nearest neighbors, we use back-translation techniques to generate paraphrases of an unlabeled sentence $\mathbf{x}_u$ in constructing $\mathbf{x'}_u$. The paraphrase $\mathbf{x}'_u$, generated via translating $\mathbf{x}_u$ to an intermediate language and then translating it back, describes the same content as $\mathbf{x}_u$ and should be close to $\mathbf{x}_u$ semantically. However, there is no tence. Inspired by the observation, we propose a new consistency loss to leverage unlabeled data: $\mathbf{x}_u$ and $\mathbf{x}'_u$ should have the same number of entities for any given entity type.

Specifically, for an unlabeled sentence $\mathbf{x}_u$ and its paraphrase $\mathbf{x}'_u$, we first guess their token labels with the current model:

$$\mathbf{y}_u = p(\mathbf{F}(\mathbf{x}_u; \theta); \phi).$$

To avoid predictions being too uniform at the early stage, we sharpen every token prediction $y_{u,i} \in \mathbf{y}_u$ with a temperature T:

$$\hat{y}_{u,i} = \frac{(y_{u,i})^{\frac{1}{T}}}{\left\|(y_{u,i})^{\frac{1}{T}}\right\|_1},$$

where $||.||_1$ denotes the $l1$-norm. We then add the prediction $\hat{y}_{u,i}$ over all tokens in the sentence to denote its total number of entities for each type:

$$\hat{y}_{u,\,\text{num}} = \sum_{i=1}^{n} \hat{y}_{u,i}.$$

Note that $\hat{y}_{u,\text{num}}$ is the guessed label vector with C-dimensions, where C is the total number of entity types. The i-th element in the $\hat{y}_{u,\text{num}}$ denotes the total number i-type entity in the sentence.

Semi-supervised LADA

During training, we use the same procedure to get the number of entities for original and each paraphrase sentence (without sharpening). Assume there are K paraphrases, denote the entity number vector for the k-the paraphrase as $\hat{y}'^{k}_{u,\text{num}}$. The consistency objective for unlabeled sentence $\mathbf{x}$ and its paraphrases is:

$$L_u = ||\hat{y}_{u,\text{num}} - \hat{y}'^{k}_{u,\text{num}}||^2. \quad (5)$$

Here we treat $\hat{y}_{u,\text{num}}$ as fixed and back-propagate only through $\hat{y}'_{u,\text{num}}$ to train the model.

Taking into account the loss objectives for both labeled and unlabeled data (Equation 1 and Equation 5), our **Semi-LADA** training objective is:

$$L_{semi} = L + \gamma L_u$$

where γ controls the trade-off between the supervised loss term and the unsupervised loss term.

Dataset

Dataset	CoNLL	GermEval
Train	14,987	24,000
Dev	3,466	2,200
Test	3,684	5,100
Entity Types	4	12
Max Sent Length	142	84

Table 2: Data statistics and our data split following Benikova et al. (2014).

In the fully supervised setting, we followed the standard data splits shown in Table 2. In the semi-supervised setting, we sampled 10,000 sentences in the training set as the unlabeled training data. We adopted FairSeq[1] to implement the back translation. For CoNLL dataset, we utilized German as the intermediate language and English as the intermediate language for GermEval.

Experiments

Model	Unlabeled data	CoNLL			GermEval		
		5%	10%	30%	5%	10%	30%
Flair (Akbik et al., 2019)	no	79.32	86.31	89.96	66.54	67.92	74.11
Flair + Intra-LADA†	no	-	-	-	-	-	-
Flair + Inter-LADA†	no	80.84	86.33	**90.61**	67.40	70.02	74.63
BERT (Devlin et al., 2019)	no	83.28	86.85	89.28	79.64	80.92	82.87
BERT + Intra-LADA†	no	83.52	87.54	89.31	79.93	81.10	82.92
BERT + Inter-LADA†	no	84.60	87.81	89.68	80.13	**81.28**	83.63
BERT + Intra&Inter-LADA†	no	**84.85**	**87.85**	89.87	**80.17**	81.23	**83.65**
VSL-GG-Hier (Chen et al., 2018)	yes	83.38	84.71	85.52	-	-	-
MT + Noise (Lakshmi Narayan et al., 2019)	yes	82.60	83.47	84.88	-	-	-
BERT + Semi-Intra-LADA†	yes	**87.15**	88.70	89.69	80.95	81.52	83.46
BERT + Semi-Inter-LADA†	yes	86.51	88.53	90.00	**81.20**	81.70	83.53
BERT + Semi-Intra&Inter-LADA†	yes	86.33	**88.78**	**90.25**	81.07	**81.77**	**83.63**

Table 3: The F1 scores on CoNLL 2003 and GermEval 2014 training with varying amounts of the labeled training data (5%, 10%, and 30% of the original training set). There were 10,000 unlabeled data for each dataset which was randomly sampled from the original training set. All the results were averaged over 5 runs. † denotes our methods.

Experiments

Model	Setting	CoNLL	GermEval
Flair (Akbik et al., 2019)	Token Classification	92.03	76.92
Flair + Intra-LADA ‡	Token Classification	-	-
Flair + Inter-LADA ‡	Token Classification	**92.12**	**78.45**
BERT (Devlin et al., 2019)	Token Classification	91.19	86.12
BERT + Intra-LADA ‡	Token Classification	91.22	86.16
BERT + Inter-LADA ‡	Token Classification	**91.83**	**86.45**
CVT (Clark et al., 2018)	Multi-task Learning	92.60	-
BERT-MRC (Li et al., 2020)	Reading Comprehension	93.04	-

Table 4: The F1 score on CoNLL 2003 and GermEval 2014 training with all the labeled training data. ‡ means incorporating our LADA data augmentation techniques into pre-trained models.

Experiments

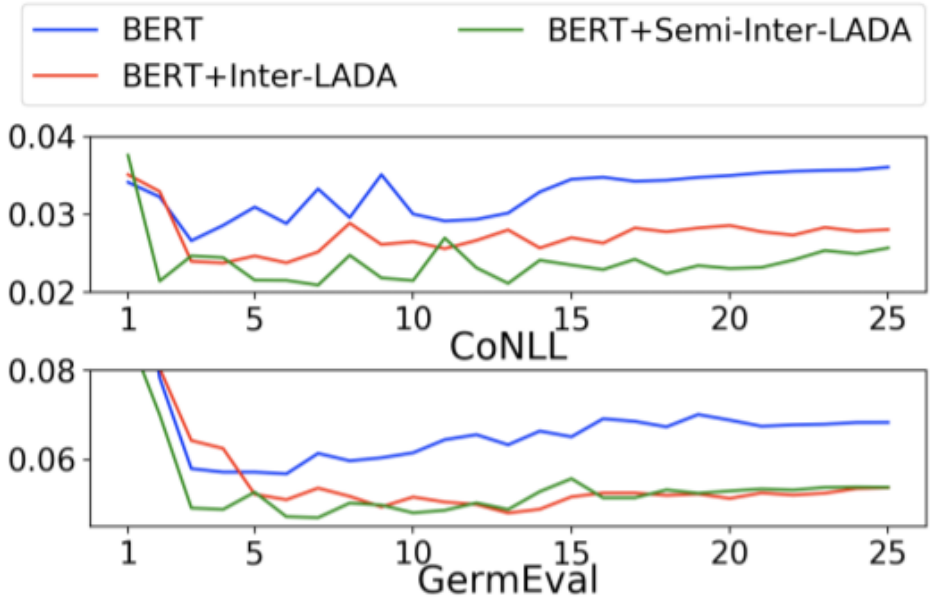

Figure 3: Loss (Y axis) on development set, trained with 5% labeled data, over different epochs (X axis).

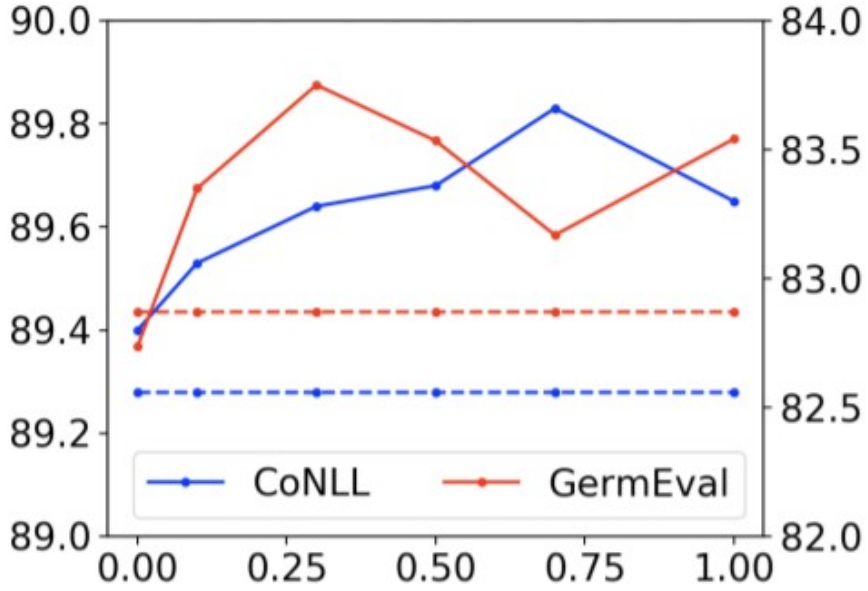

Figure 4: F1 score on test set training with 30% labeled data with different μ in BERT + Inter-LADA. The left Y axis is for CoNLL, and the right Y axis is for GermEval. Dashed lines are the F1 scores of BERT model.