MixText: Linguistically-Informed Interpolation of Hidden Space for Semi-Supervised Text Classification

Jiaao Chen
Georgia Tech
jchen896@gatech.edu

Zichao Yang
CMU
zichaoy@cs.cmu.edu

Diyi Yang
Georgia Tech
dyang888@gatech.edu

Limited labeled data

- Task: semi-supervised text classification

- Tmix
- MixText

Tmix

- Mixup can be viewed a data augmentation approach which creates new data samples based on the original training set.
- it enforces a regularization on the model to behave linearly among the training data.

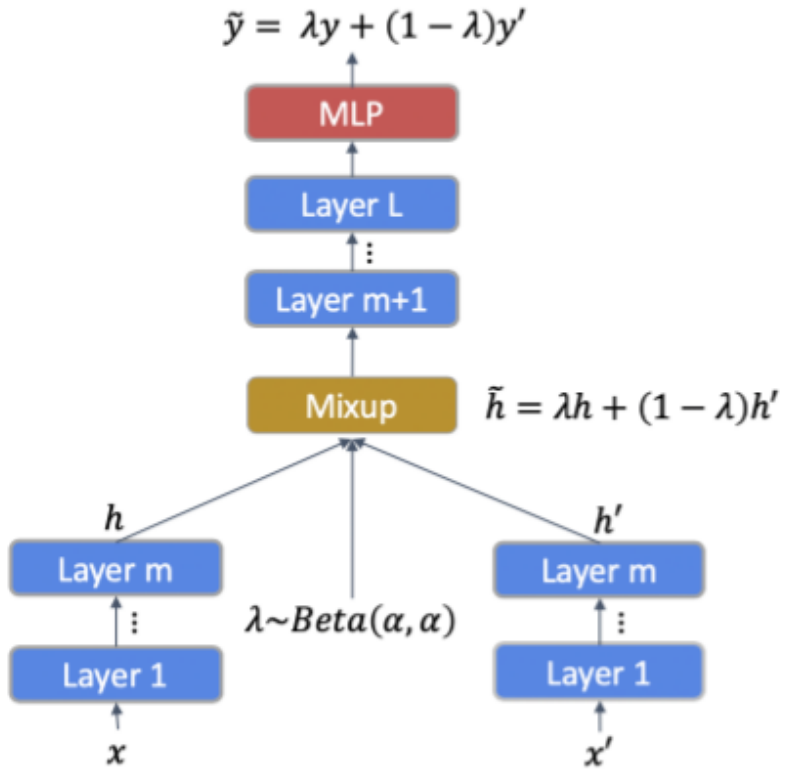

Figure 1: TMix takes in two text samples x and x' with labels y and y', mixes their hidden states h and h' at layer m with weight λ into $\tilde{h}$, and then continues forward passing to predict the mixed labels $\tilde{y}$.

Tmix

- The main idea of Mixup is very simple: given two labeled data points (xi,yi) and (xj,yj), where x can be an image and y is the one-hot representation of the label

$$\tilde{\mathbf{x}} = \text{mix}(\mathbf{x}_i, \mathbf{x}_j) = \lambda \mathbf{x}_i + (1-\lambda)\mathbf{x}_j, \quad (1)$$

$$\tilde{\mathbf{y}} = \text{mix}(\mathbf{y}_i, \mathbf{y}_j) = \lambda \mathbf{y}_i + (1-\lambda)\mathbf{y}_j, \quad (2)$$

where $\lambda \in [0, 1]$. The new virtual training samples are used to train a neural network model. Mixup can be interpreted in different ways. On

Tmix

- we first compute the hidden representations of two text samples separately in the bottom layers. Then we mix up the hidden representations at layer m, and feed the interpolated hidden representations to the upper layers.

$$\mathbf{h}_l^i = g_l(\mathbf{h}_{l-1}^i; \boldsymbol{\theta}), l \in [1, m],$$
$$\mathbf{h}_l^j = g_l(\mathbf{h}_{l-1}^j; \boldsymbol{\theta}), l \in [1, m].$$

The mixup at the m-th layer and continuing forward passing to upper layers are defined as:

$$\tilde{\mathbf{h}}_m = \lambda \mathbf{h}_m^i + (1 - \lambda)\mathbf{h}_m^j,$$
$$\tilde{\mathbf{h}}_l = g_l(\tilde{\mathbf{h}}_{l-1}; \boldsymbol{\theta}), l \in [m + 1, L].$$

We call the above method **TMix** and define the new mixup operation as the whole process to get $\tilde{\mathbf{h}}_L$:

$$\text{TMix}(\mathbf{x}_i, \mathbf{x}_j; g(.; \boldsymbol{\theta}), \lambda, m) = \tilde{\mathbf{h}}_L.$$

Tmix

In our experiments, we sample the mix parameter λ from a Beta distribution for every batch to perform the interpolation :

$$\lambda \sim \text{Beta}(\alpha, \alpha),$$
$$\lambda = \max(\lambda, 1 - \lambda),$$

in which α is the hyper-parameter to control the distribution of λ. In TMix, we mix the labels in

Building on those findings, we choose the layers that contain both syntactic and semantic information as our mixing layers, namely $\mathbf{M} = \{7, 9, 12\}$. For every batch, we *randomly sample* m, the layer to mixup representations, from the set $\mathbf{M}$ computing the interpolation. We also performed ablation

$$L_{\text{TMix}} = \text{KL}(\text{mix}(\mathbf{y}_i, \mathbf{y}_j)||p(\text{TMix}(\mathbf{x}_i, \mathbf{x}_j); \boldsymbol{\phi})$$

where $p(.; \boldsymbol{\phi})$ is a classifier on top of the encoder model. In our experiments, we implement the classifier as a two-layer MLP, which takes the mixed representation $\text{TMix}(\mathbf{x}_i, \mathbf{x}_j)$ as input and returns a probability vector. We jointly optimize over the encoder parameters $\boldsymbol{\theta}$ and the classifier parameters $\boldsymbol{\phi}$ to train the whole model.

Semi-supervised MixText

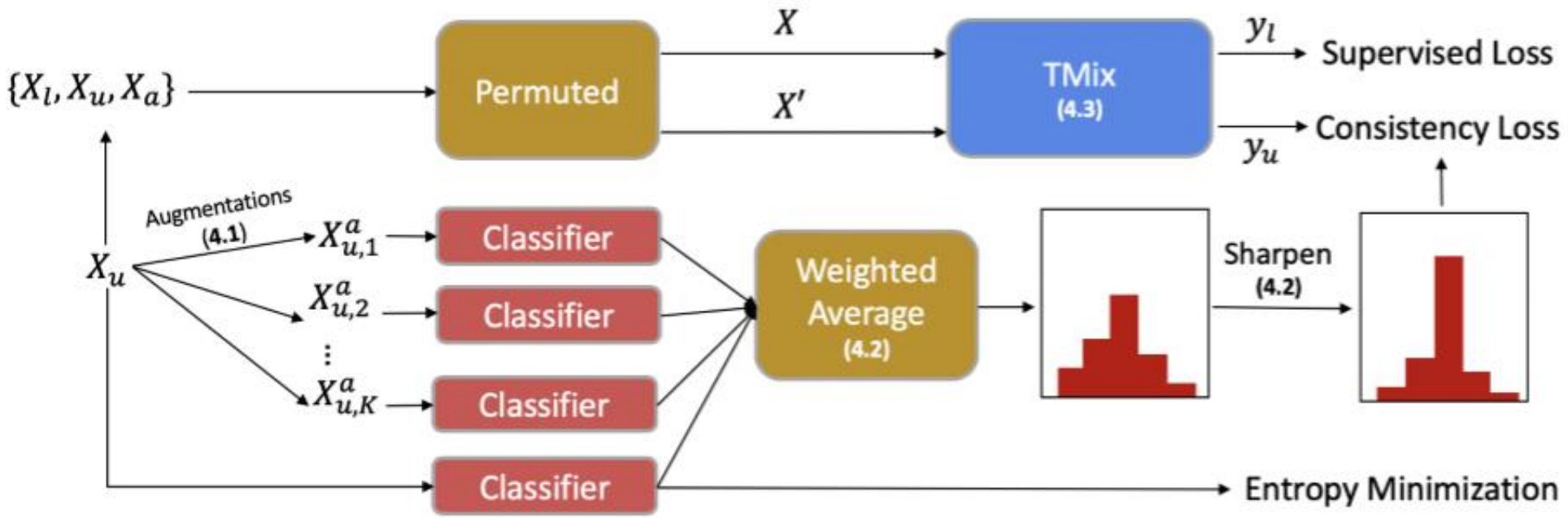

Figure 2: Overall Architecture of MixText. MixText takes in labeled data and unlabeled data, conducts augmentations and predicts labels for unlabeled data, performs TMix over labeled and unlabeled data, and computes supervised loss, consistency loss and entropy minimization term.

Semi-supervised MixText

- we come up a label guessing method to generate labels for the unlabeled data in the training process.
- With the guessed labels, we can treat the unlabeled data as additional labeled data and perform TMix for training
- Moreover, we combine TMix with additional data augmentation techniques to generate large amount of augmented data
- we introduce an entropy minimization loss that encourages the model to assign sharp probabilities on unlabeled data samples,

Data Augmentation

- Back translations
- For each xu,i in the unlabeled text set Xu, we generate K augmentations xai,k = augment k(xu,i),k ∈ [1,K] by back translations with different intermediate languages. For example, we can translate original sentences from English to German and then translate them back to get the paraphrases.

For unlabeled data, we selected German and Russian as intermediate languages for back translations using FairSeq[2], and the random sampling temperature was 0.9. Here is an example, for a news from AG News dataset: *"Oil prices rallied to a record high above $55 a barrel on Friday on rising fears of a winter fuel supply crunch and robust economic growth in China, the world's number two user"*, the augment texts through German and Russian are: *"Oil prices surged to a record high above $55 a barrel on Friday on growing fears of a winter slump and robust economic growth in world No.2 China"* and *"Oil prices soared to record highs above $55 per barrel on Friday amid growing fears over a winter reduction in U.S. oil inventories and robust economic growth in China, the world's second-biggest oil consumer"*.

Label Guessing

For an unlabeled data sample $\mathbf{x}_i^u$ and its K augmentations $\mathbf{x}_{i,k}^a$, we generate the label for them using weighted average of the predicted results from the current model:

$$\mathbf{y}_i^u = \frac{1}{w_{ori} + \sum_k w_k}(w_{ori} p(\mathbf{x}_i^u) + \sum_{k=1}^{K} w_k p(\mathbf{x}_{i,k}^a))).$$

Note that $\mathbf{y}_i^u$ is a probability vector. We expect the model to predict consistent labels for different augmentations. Hence, to enforce the constraint, we use the weighted average of all predictions, rather than the prediction of any single data sample, as the generated label. Moreover, by explicitly introducing the weight w_{ori} and w_k, we can control the contributions of different quality of augmentations

To avoid the weighted average being too uniform, we utilize a sharpening function over predicted labels. Given a temperature hyper-parameter T:

$$\text{Sharpen}(\mathbf{y}_i^u, T) = \frac{(\mathbf{y}_i^u)^{\frac{1}{T}}}{||(\mathbf{y}_i^u)^{\frac{1}{T}}||_1},$$

where $||.||_1$ is l_1-norm of the vector. When $T \to 0$, the generated label becomes a one-hot vector.

Tmix on Labeled and Unlabeled Data

After getting the labels for unlabeled data, we merge the labeled text $\mathbf{X}_l$, unlabeled text $\mathbf{X}_u$ and unlabeled augmentation text $\mathbf{X}_a = \{x^a_{i,k}\}$ together to form a super set $\mathbf{X} = \mathbf{X}_l \cup \mathbf{X}_u \cup \mathbf{X}_a$. The corresponding labels are $\mathbf{Y} = \mathbf{Y}_l \cup \mathbf{Y}_u \cup \mathbf{Y}_a$, where $\mathbf{Y}^a = \{\mathbf{y}^a_{i,k}\}$ and we define $\mathbf{y}^a_{i,k} = \mathbf{y}^u_i$, i.e., the all augmented samples share the same generated label as the original unlabeled sample.

In training, we randomly sample two data points $\mathbf{x}, \mathbf{x}' \in \mathbf{X}$, then we compute $\mathrm{TMix}(\mathbf{x}, \mathbf{x}')$, $\mathrm{mix}(\mathbf{y}, \mathbf{y}')$ and use the KL-divergence as the loss:

$$L_{\mathrm{TMix}} = \mathbb{E}_{\mathbf{x},\mathbf{x}' \in \mathbf{X}} \mathrm{KL}(\mathrm{mix}(\mathbf{y}, \mathbf{y}') || p(\mathrm{TMix}(\mathbf{x}, \mathbf{x}')).$$

Since $\mathbf{x}, \mathbf{x}'$ are randomly sampled from $\mathbf{X}$, we interpolate text from many different categories: mixup among among labeled data, mixup of labeled and unlabeled data and mixup of unlabeled data. Based on the categories of the samples, the loss can be divided into two types:

Supervised loss When $\mathbf{x} \in \mathbf{X}_l$, the majority information we are actually using is from the labeled data, hence training the model with supervised loss.

Consistency loss When the samples are from unlabeled or augmentation set, i.e., $\mathbf{x} \in \mathbf{X}^u \cup \mathbf{X}^a$, most information coming from unlabeled data, the KL-divergence is a type of consistency loss, constraining augmented samples to have the same labels with the original data sample.

Semi-supervised MixText

4.4 Entropy Minimization

To encourage the model to produce confident labels on unlabeled data, we propose to minimize the entropy of prediction probability on unlabeled data as a self-training loss:

$$L_{\text{margin}} = \mathbb{E}_{\mathbf{x} \in \mathbf{X}_u} \max(0, \gamma - ||\mathbf{y}^u||_2^2),$$

where γ is the margin hyper-parameter. We minimize the entropy of the probability vector if it is larger than γ.

Combining the two losses, we get the overall objective function of MixText:

$$L_{\text{MixText}} = L_{\text{TMix}} + \gamma_m L_{\text{margin}}.$$

Experiment

- Baseline

ForTMix, we only utilize the labeled dataset as the settings in Bert baseline, and set the batch size as 8. InMixText, we utilize both labeled data and unlabeled data for training using the same settings as in UDA. We set K = 2, i.e., for each unlabeled data we perform two augmentations, specifically German and Russian.

- **VAMPIRE** (Gururangan et al., 2019): VAriational Methods for Pretraining In Resource-limited Environments(VAMPIRE) pretrained a unigram document model as a variational autoencoder on in-domain, unlabeled data and used its internal states as features in a downstream classifier.
- **BERT** (Devlin et al., 2019): We used the pretrained BERT-based-uncased model[3] and fine-tuned it for the classification. In details, we used average pooling over the output of BERT encoder and the same two-layer MLP as used in MixText to predict the labels.
- **UDA** (Xie et al., 2019): Since we do not have access to TPU and need to use smaller amount of unlabeled data, we implemented Unsupervised Data Augmentation(UDA) using pytorch by ourselves. Specifically, we used the same BERT-based-uncased model, unlabeled augment data and batch size as our MixText, used original unlabeled data to predict the labels with the same softmax sharpen temperature as our MixText and computed consistency loss between augmented unlabeled data.

Experiment

Dataset	Label Type	Classes	Unlabeled	Dev	Test
AG News	News Topic	4	5000	2000	1900
DBpedia	Wikipeida Topic	14	5000	2000	5000
Yahoo! Answer	QA Topic	10	5000	5000	6000
IMDB	Review Sentiment	2	5000	2000	12500

Table 1: Dataset statistics and dataset split. The number of unlabeled data, dev data and test data in the table means the number of data per class.

Datset	Model	10	200	2500	Dataset	Model	10	200	2500
AG News	VAMPIRE	-	83.9	86.2	DBpedia	VAMPIRE	-	-	-
	BERT	69.5	87.5	90.8		BERT	95.2	98.5	99.0
	TMix*	74.1	88.1	91.0		TMix*	96.8	98.7	99.0
	UDA	84.4	88.3	91.2		UDA	97.8	98.8	99.1
	MixText*	**88.4**	**89.2**	**91.5**		MixText*	**98.5**	**98.9**	**99.2**
Yahoo!	VAMPIRE	-	59.9	70.2	IMDB	VAMPIRE	-	82.2	85.8
	BERT	56.2	69.3	73.2		BERT	67.5	86.9	89.8
	TMix*	58.6	69.8	73.5		TMix*	69.3	87.4	90.3
	UDA	63.2	70.2	73.6		UDA	78.2	89.1	90.8
	MixText*	**67.6**	**71.3**	**74.1**		MixText*	**78.7**	**89.4**	**91.3**

Table 2: Performance (test accuracy(%)) comparison with baselines. The results are averaged after three runs to show the significance (Dror et al., 2018), each run takes around 5 hours. Models are trained with 10, 200, 2500 labeled data per class. VAMPIRE, Bert, and TMix do not use unlabeled data during training while UDA and MixText utilize unlabeled data. * means our models.

Experiment

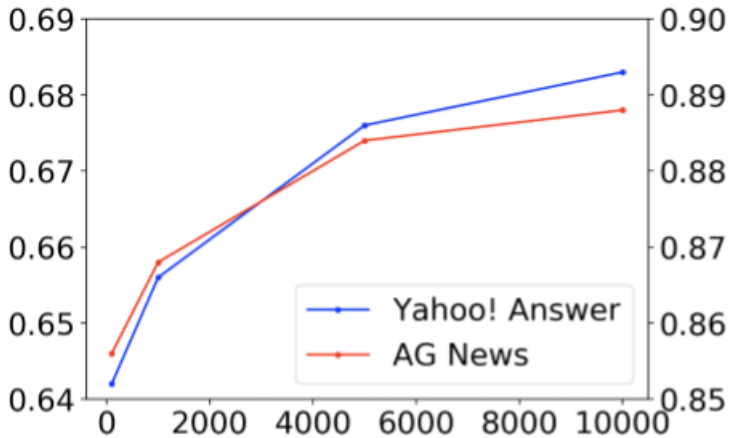

Figure 4: Performance (test accuracy (%)) on AG News (y axis on the right) and Yahoo! Answer (y axis on the left) with 10 labeled data and varying number of unlabeled data per class for MixText.

Mixup Layers Set	Accuracy(%)
∅	69.5
{0,1,2}	69.3
{3,4}	70.4
{6,7,9}	71.9
{7,9,12}	**74.1**
{6,7,9,12}	72.2
{3,4,6,7,9,12}	71.6

Table 3: Performance (test accuracy (%)) on AG News with 10 labeled data per class with different mixup layers set for TMix. ∅ means no mixup.

Model	Accuracy(%)
MixText	**67.6**
- weighted average	67.1
- TMix	63.5
- unlabeled data	58.6
- all	56.2

Table 4: Performance (test accuracy (%)) on Yahoo! Answer with 10 labeled data and 5000 unlabeled data per class after removing different parts of MixText.