Contrast-enhanced Semi-supervised Text Classification with Few Labels

Austin Cheng-Yun Tsai, Sheng-Ya Lin, Li-Chen Fu
Department of Computer Science and Information Engineering, National Taiwan University
{r08922086, r09944044, lichen}@ntu.edu.tw

2022.9.11 • ChongQing

—— AAAI 2022

Reported by JiaWei Cheng

1.Introduction

2.Overview

3.Methods

4.Experiments

Introduction

1.When there is only a limited number of labeled data, complex neural networks often suffer from over-fitting

2. Semi-supervised approaches typically involve a sophisticated Neural Machine Translation (NMT) system,such approaches may be bothersome in real-world scenarios by requiring an additional NMT system

3.Traditional self-training does not perform sample selection, nor does it consider noises in the generated pseudo-labels during training

Overview

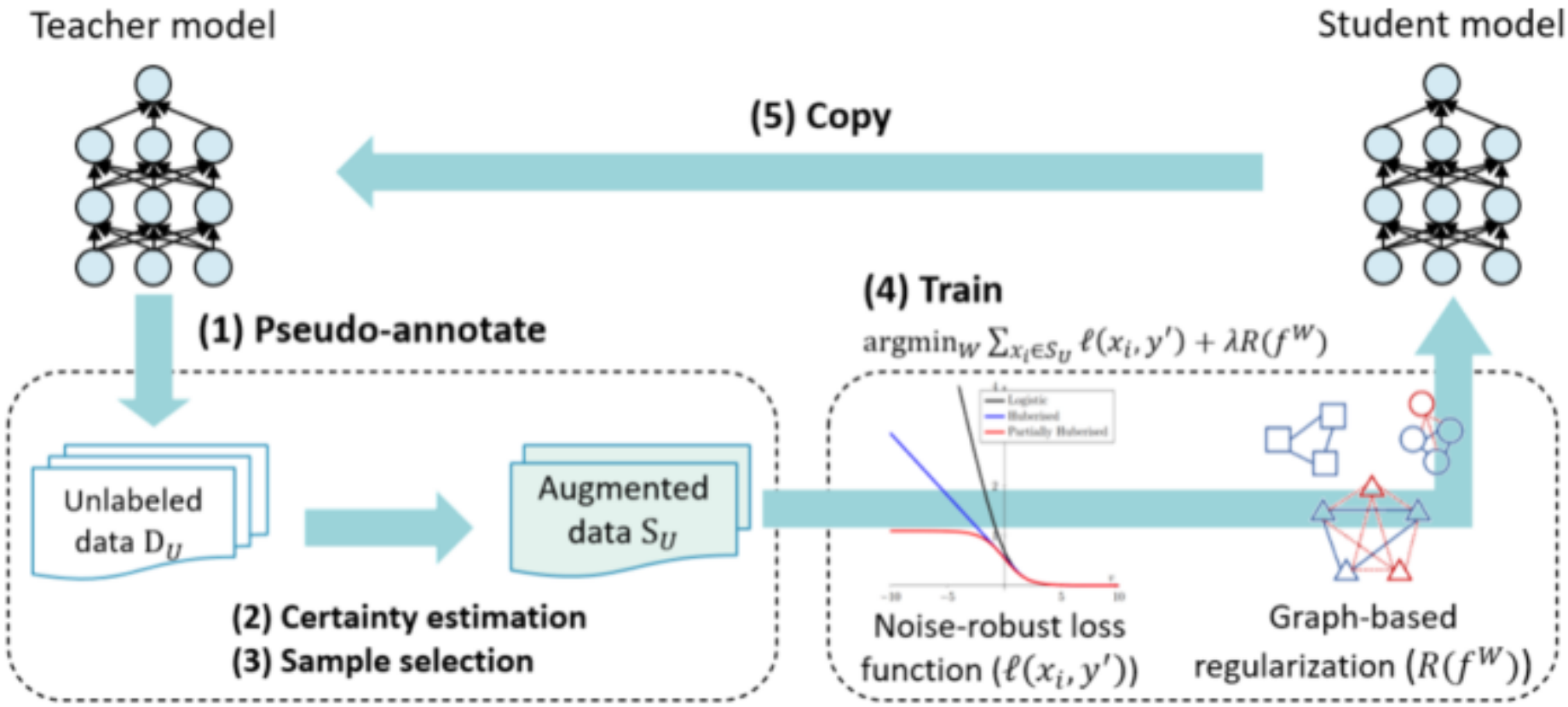

Figure 1: Framework overview.

Method

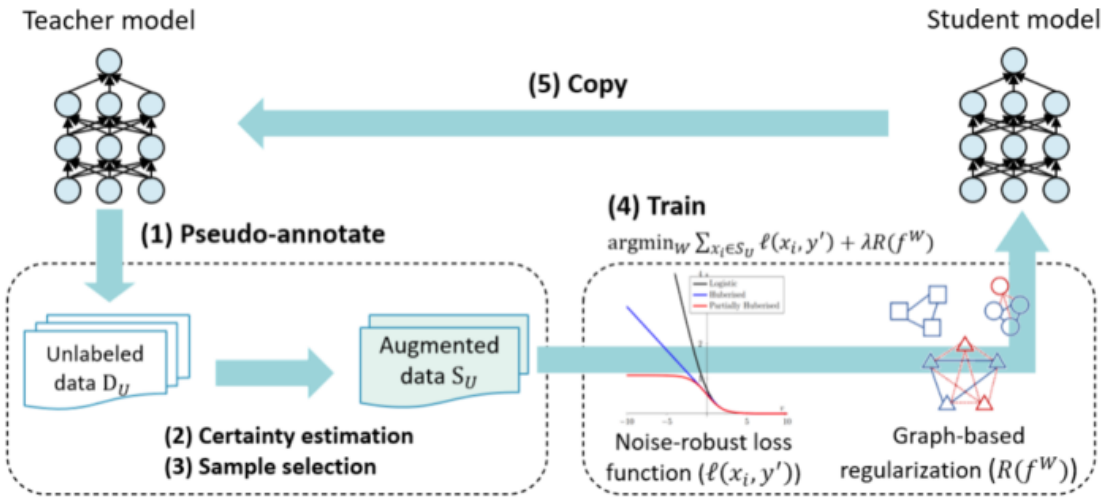

Figure 1: Framework overview.

$$\underset{W}{\text{argmin}} \sum_{x_i \in D_L \cup D_U} \ell(x_i, y') + \lambda R(f^W) \qquad (1)$$

where y' is the target label. If x_i is from D_U, y' is defined as the predicted label (pseudo-label). $\ell(\cdot, \cdot) : \mathbb{Y} \times \mathbb{Y} \to \mathbb{R}$ is the loss function measuring the classification loss between the predictions and the labels, typically the cross-entropy loss. R is the regularization term that prevents the model from overly aggressively learning from data. λ is the hyper-parameter controlling the impact of R.

Method

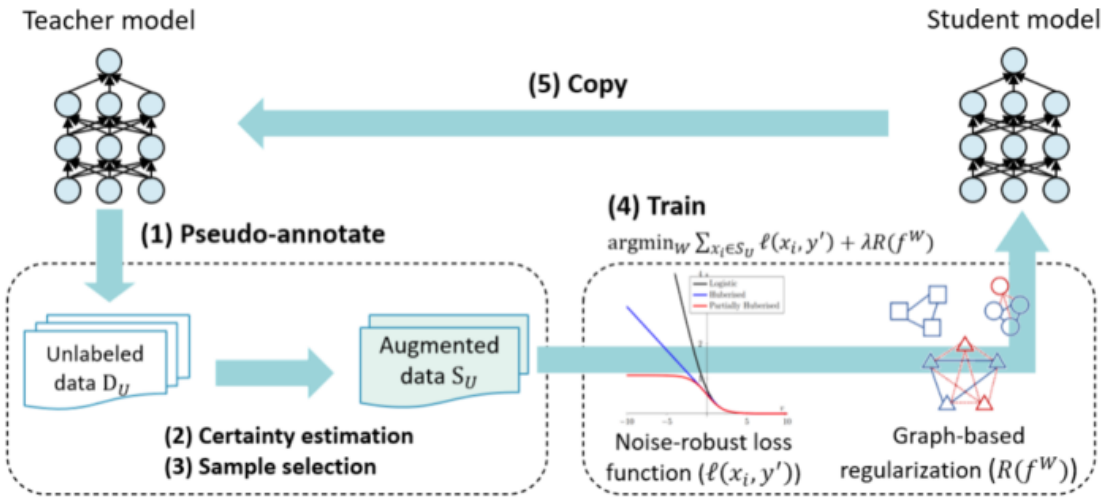

Figure 1: Framework overview.

$$\underset{W}{\text{argmin}} \sum_{x_i \in S_U, S_U \subseteq D_U} \ell(x_i, \widetilde{y}_i) + \lambda R(W) \qquad (2)$$

where W is the model parameters for the *student* model, and the hard pseudo-labels $\widetilde{y}_i$ are given by the *teacher* model W^* from the last iteration (W^* will be fixed in current iteration):

$$\widetilde{y}_i = \underset{c}{\text{argmax}}\, p(y = c | f^{W^*}(x)) \qquad (3)$$

Method

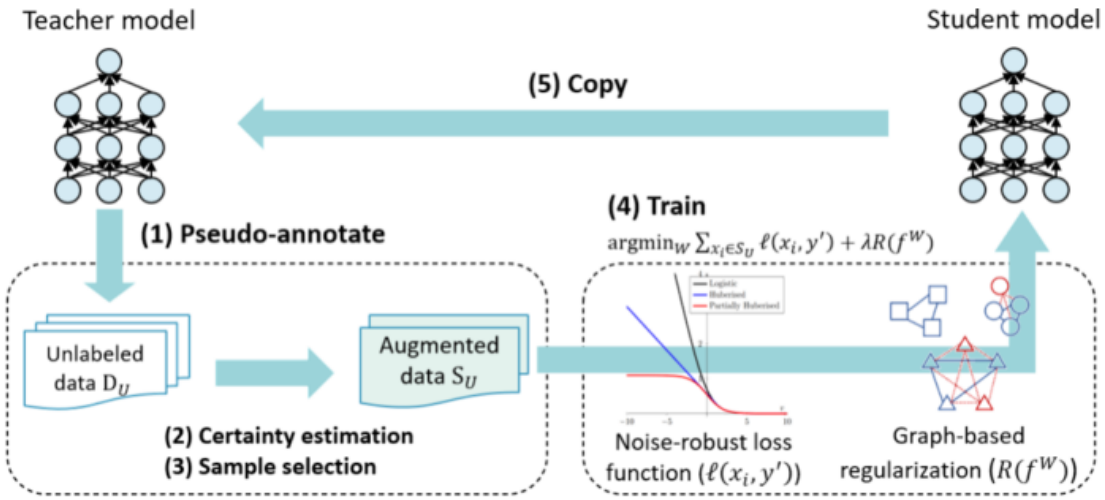

Figure 1: Framework overview.

Instead of having deterministic weights, BNN assumes a prior distribution over its model parameters. Considering the mapping function f^W for BNN, where W is the model parameters, the parameter optimization is achieved by finding the posterior distribution over model parameters $p(W|D_{train})$ on a training dataset D_{train}. During inference, for data instance x, the probability for class c is $p(y = c|x) = \int_W p(y = c|f^W(x))p(W|D_{train})dW$. However, it is computationally intractable to calculate over all possible W and we have to find a surrogate distribution $q_\theta(W)$ in a tractable family of distributions to replace the true model posterior $p(W|D_{train})$. Gal and Ghahramani (2016) and Gal, Islam, and Ghahramani (2017) developed Monte-Carlo Dropout (MC Dropout) using BNN and showed that the probability for class c, $p(y = c|x)$, can be approximated by considering $q_\theta(W)$ to be the dropout distribution (Srivastava et al. 2014), which is tractable, with T masked model weights $\{\widetilde{W}_t\}_{t=1}^T \sim q_\theta(W)$:

$$p(y = c|x, D_{train}) \approx \frac{1}{T} \sum_{t=1}^{T} p(y = c|f^{\widetilde{W}_t}(x)) \qquad (4)$$

Method

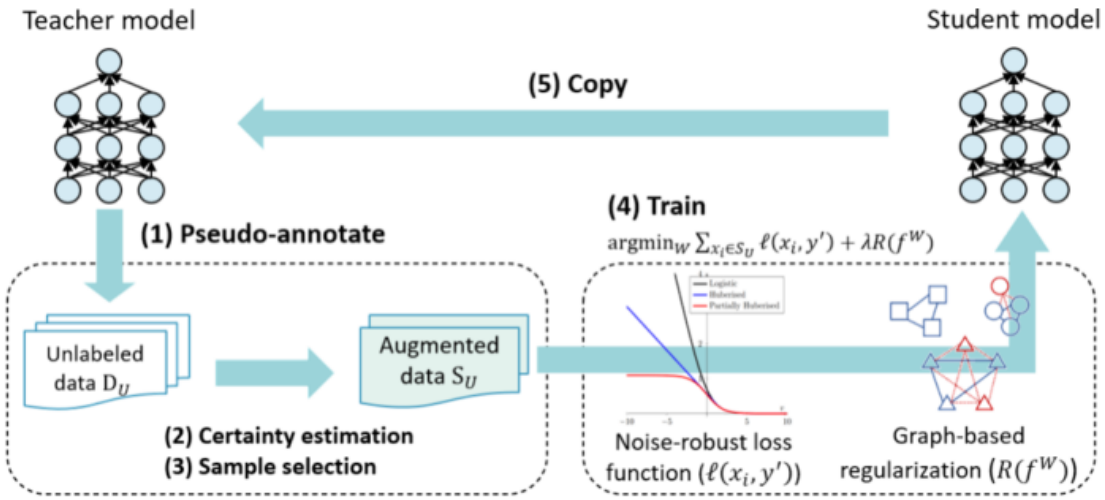

Figure 1: Framework overview.

where $\hat{p}_c^t = p(y = c|f^{\widetilde{W}_t(x)})$ is the estimated probability of class c given by the model parameters $\widetilde{W}_t \sim q_\theta(W)$ in the t-th trial in the MC Dropout

Using entropy $\mathbb{H}(\cdot)$ to measure the level of information we have, we define the information gain $\mathbb{B}$ to be the difference between the final entropy $\mathbb{H}(y|x, D_U)$ after seeing the whole unlabeled dataset D_U and the current entropy $\mathbb{H}(y|x, W)$ given the model parameters W in the current iteration. Formally, for data sample $x \in D_U$, the information gain $\mathbb{B}$ with respect to its expected label is

$$\mathbb{B}(y, W|x, D_U) = \mathbb{H}(y|x, D_U) - \mathbb{E}_{p(W|D_U)}[\mathbb{H}(y|x, W)] \tag{5}$$

where $p(W|D_U)$ is the posterior distribution of the model parameter in current iteration. As Eq. (5) is computationally intractable, it can be approximated by MC Dropout (Gal, Islam, and Ghahramani 2017):

$$\widehat{\mathbb{B}}(y, W|x, D_U) = -\sum_c (\frac{1}{T}\sum_t \hat{p}_c^t) \log (\frac{1}{T}\sum_t \hat{p}_c^t) + \frac{1}{T}\sum_t \sum_c \hat{p}_c^t \log (\hat{p}_c^t) \tag{6}$$

Method

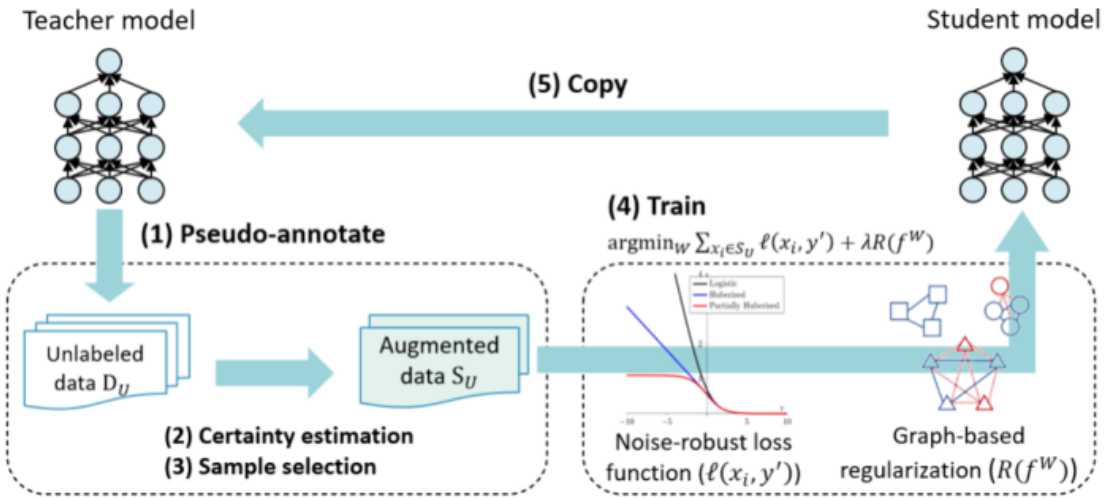

Figure 1: Framework overview.

$$s_i = \frac{1 - \hat{\mathbb{B}}(y_i, W | x_i, D_U)}{\sum_{x_j \in D_U} [1 - \hat{\mathbb{B}}(y_j, W | x_j, D_U)]} \tag{7}$$

,where $\sum_{x_j \in D_U} [1 - \hat{\mathbb{B}}(y_j, W | x_j, D_U)]$ is a normalizing factor

Method

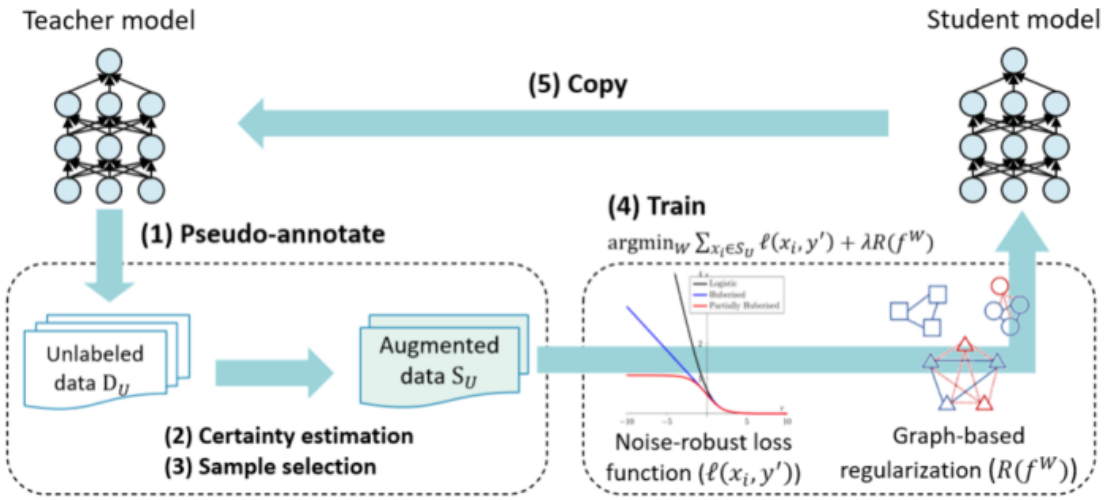

Figure 1: Framework overview.

$$V = \{i \mid x_i \in S_U\}$$
$$E = \{(i,j) \mid x_i, x_j \in S_U\} \tag{8}$$

and edge (i, j) has edge weight satisfying the following:

$$e_{ij} = \begin{cases} 1, & \text{if } \tilde{y}_i = \tilde{y}_j \\ 0, & \text{if } \tilde{y}_i \neq \tilde{y}_j \end{cases} \tag{9}$$

where $\widetilde{y}_i, \widetilde{y}_j$ are the hard pseudo-labels of x_i obtained by the teacher model W^* from the last iteration.

Method

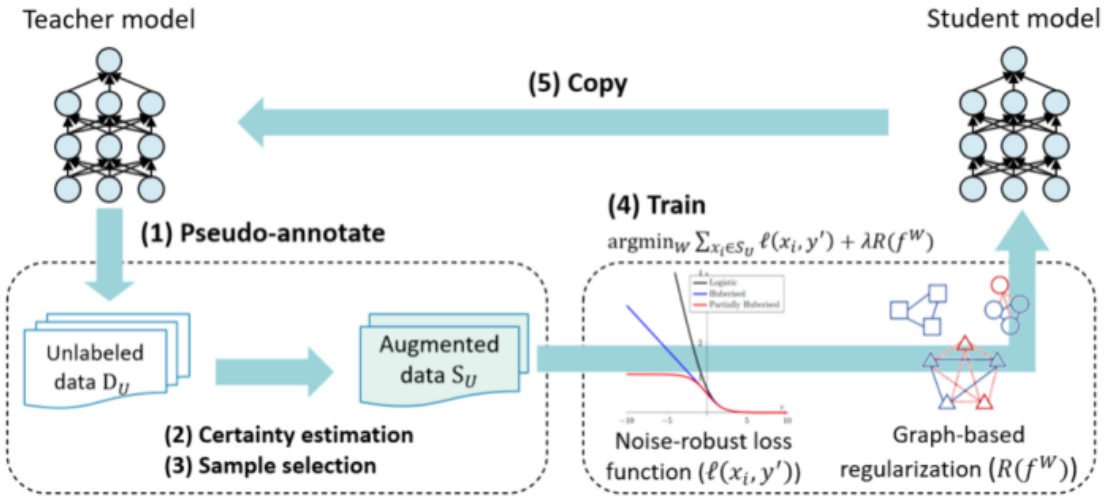

Figure 1: Framework overview.

Mathe-matically, given a similarity graph G and a pre-defined mar-gin m between dissimilar features, we enhance the contrastthrough the regularization term R in the training objective:

$$R(f^W, G) = \sum_{(i,j)\in E, e_{ij}=1} ||h(x_i) - h(x_j)||_2^2 + \sum_{(i,j)\in E, e_{ij}=0} \max(0, m - ||h(x_i) - h(x_j)||_2)^2 \quad (10)$$

Method

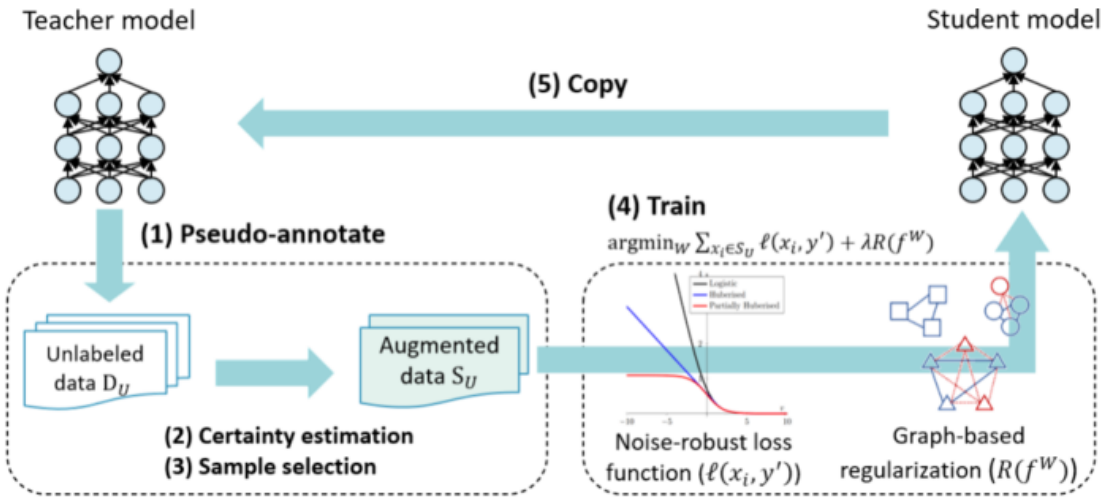

Figure 1: Framework overview.

$$\gamma(v_i) = 1 - \texttt{Var}(y = \tilde{y}_i | x_i) \approx 1 - \frac{1}{T}\sum_{t=1}^{T} p_t^2 + (\frac{1}{T}\sum_{t=1}^{T} p_t)^2 \tag{11}$$

$$\gamma(e_{ij}) = \frac{\gamma(v_i) + \gamma(v_j)}{2} \tag{12}$$

where $p_t = p(y = \tilde{y}_i | f^{\widetilde{W}_t}(x_i))$ is the probability of $\tilde{y}_i$ predicted by $f^{\widetilde{W}_t}$, where $\widetilde{W}_t$ is the model weight sampled in the t-th iteration in MC Dropout.

Method

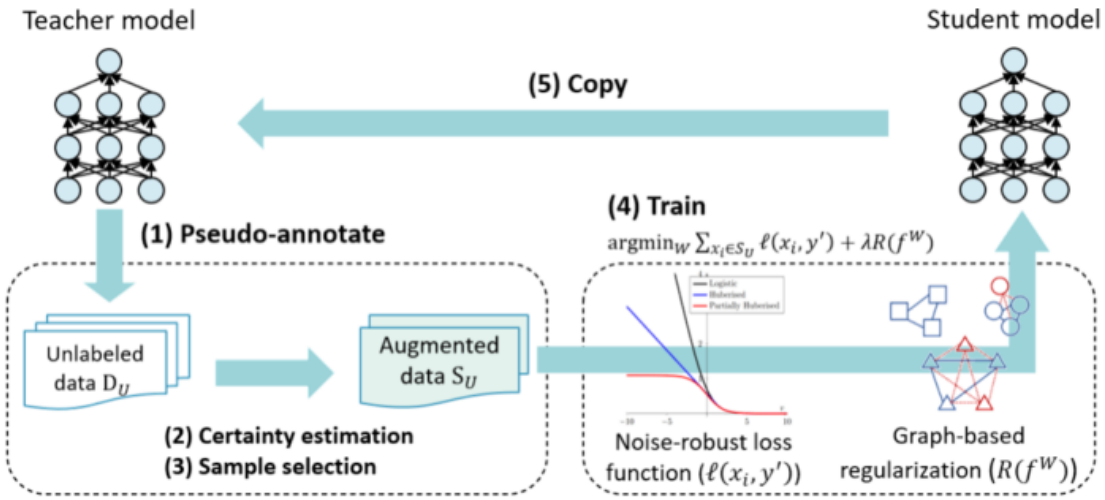

Figure 1: Framework overview.

To construct the reliable sub-graph $G'(V', E')$, we add all the nodes in G into G', *i.e.* $V' = V$. As for the edge set E', for each node v_i, we select the top k edges with the highest reliability γ from the positive set and from the negative set to form the reliable edge set E' ($2k$ edges in total for each node), where k is a hyper-parameter, and the positive set and the negative set are defined as

1. positive set: $\{(i,j) \mid e_{ij} = 1, j = 1, 2, ..., |S_U|\}$
2. negative set: $\{(i,j) \mid e_{ij} = 0, j = 1, 2, ..., |S_U|\}$

In this way, we have a reliable graph with much fewer wrong edges, and we can still perform the same optimization on the resulting sub-graph G' as before:

$$R(f^W, G') = \sum_{(i,j)\in E', e_{ij}=1} ||h(x_i) - h(x_j)||_2^2 + \sum_{(i,j)\in E', e_{ij}=0} \max(0, m - ||h(x_i) - h(x_j)||_2)^2 \quad (13)$$

Method

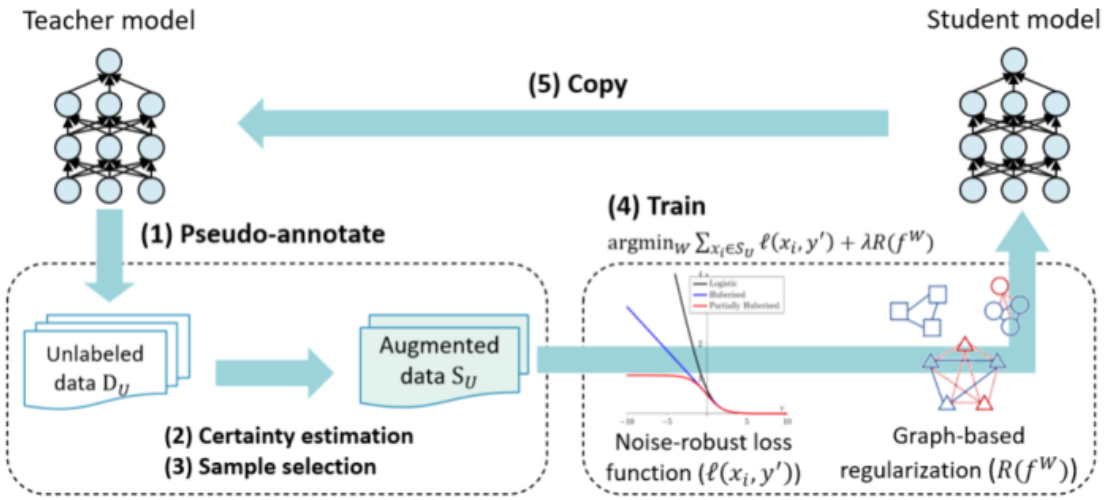

Figure 1: Framework overview.

Utilizing traditional cross-entropy loss to train pseudo-annotated data is prone to error accumulation. Consider $l(f^W(x), y) = \varphi(p_W(x, y))$ to be the loss function, where $p_W(x, y)$ is the probability of the target class y predicted by the model W, and $\varphi(u)$ is the classification loss. For cross-entropy loss, $l(f^W(x), y) = -\ln(p_W(x, y))$. Then, the gradient induced by the cross-entropy loss is

$$\frac{\partial l(f^W(x), y)}{\partial W} = -\frac{\partial \ln p_W(x, y)}{\partial W} = -\frac{1}{p_W(x, y)} \frac{\partial p_W(x, y)}{\partial W} \tag{14}$$

$$\widetilde{l} = \begin{cases} -\tau p_W(x, y) - \varphi^*(-\tau), & \text{if } \varphi'(p_W(x, y)) \leq -\tau \\ -\log p_W(x, y), & \text{otherwise} \end{cases} \tag{15}$$

Integrating it with cross-entropy loss , the partially huberised cross-entropy loss (PHCE loss) is obtained:

$$\widetilde{l} = \begin{cases} -\tau p_W(x, y) + \log \tau + 1, & \text{if } p_W(x, y) \leq \frac{1}{\tau} \\ -\log p_W(x, y), & \text{otherwise} \end{cases} \tag{16}$$

where τ is a hyper-parameter related to the degree of noise.
τ is set larger if the data are essentially noise-free

Experiments

Dataset	Full training	30 labeled training data per class				
	BERT	**BERT**	**UDA**	**Standard ST**	**UST**	**CEST (Ours)**
AG News	92.98	79.84	85.92	84.07	86.90	**87.05**
DBpedia	99.13	98.01	96.88	97.25	98.30	**98.61**
IMDB	91.26	80.90	89.30	83.81	84.06	**90.20**
Elec	96.48	85.07	89.64	89.50	89.97	**92.26**
SST-2	93.19	74.23	83.58	84.81	88.09	**89.71**
Average	94.61	83.61 (+0.00%)	89.06 (+5.45%)	87.89 (+4.28%)	89.46 (+5.85%)	**91.57** (+7.96%)

Table 2: Performance (test accuracy(%)) comparison with baselines. The results are averaged for three runs, with each run taking 3-8 hours on an NVIDIA RTX3090. BERT on the second column means directly fine-tuning without using unlabeled data. Standard ST denotes standard self-training. We reproduced all baselines with PyTorch except that UDA's results are cited.

Experiments

Figure 3: t-SNE visualization. (a) without similarity graph (b) similarity graph without considering reliability (c) with reliable similarity graph.

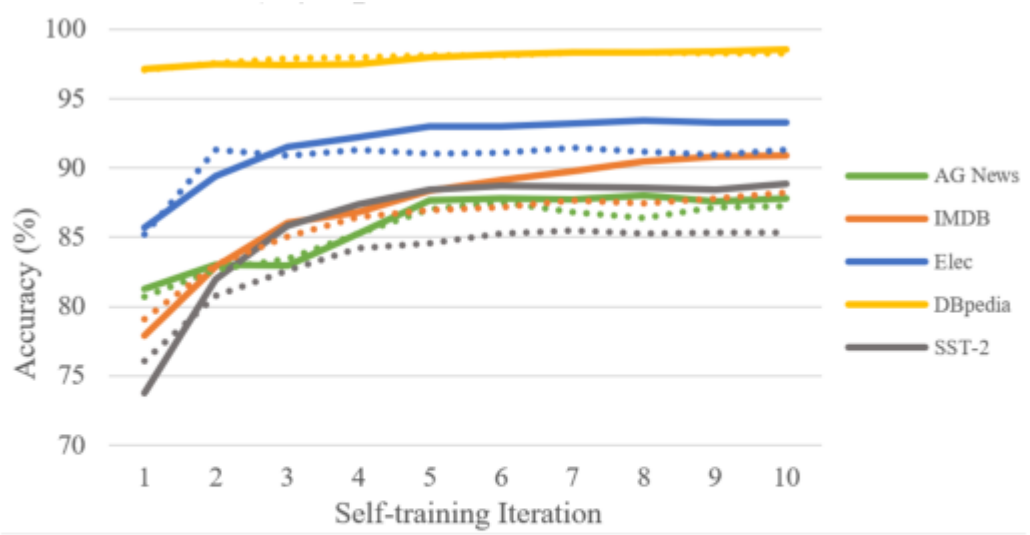

Figure 4: Test accuracy (%) over self-training iterations. The solid and the dotted lines shows the results of training with and without using the noise-robust loss (PHCE loss).

Experiments

	AG News	IMDB	DBpedia	Elec	SST-2	Average
BERT (direct fine-tuning)	79.84	80.90	98.01	85.07	74.23	83.61
CEST	**87.05**	**90.20**	**98.61**	**92.26**	**89.71**	**91.57**
– noise-robust loss	86.80	89.68	97.91	92.08	88.75	91.04
– reliable sub-graph	86.56	88.37	97.44	91.96	87.95	90.46
– smoothness regularization	86.10	86.76	98.18	90.47	88.33	89.97
– without sampling	84.07	83.81	97.25	89.51	84.81	87.89

Table 3: Ablation Study of performance (test accuracy (%)) over different design configurations.

Experiments

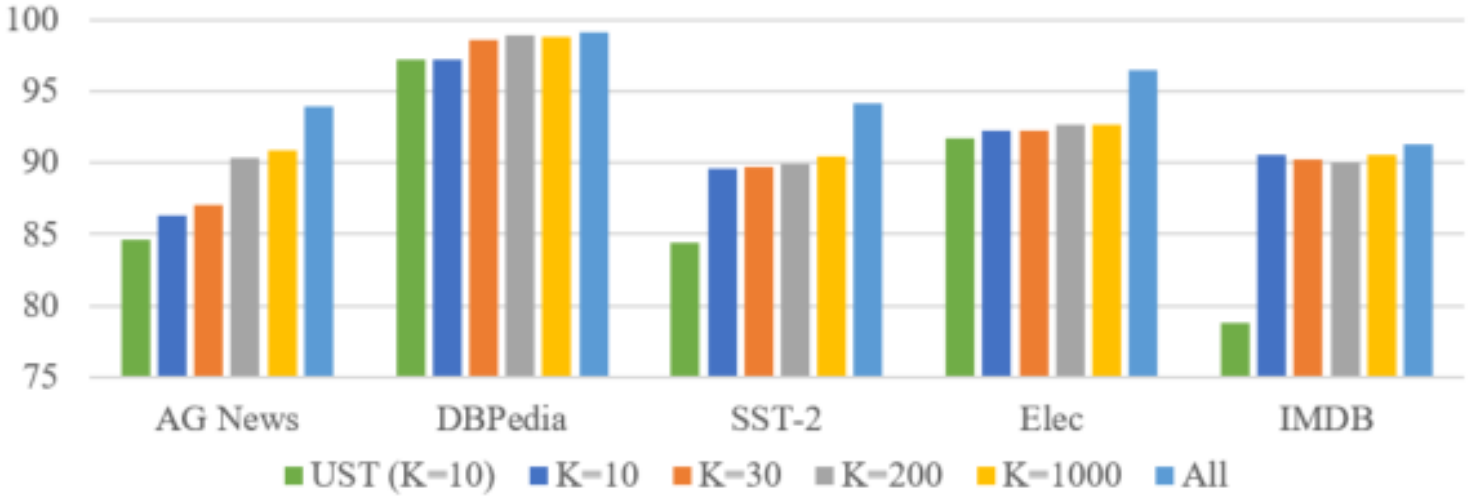

Figure 5: Test accuracy (%) under different number (K) of labeled training data per class.

Experiments

Dataset	BERT	UST	CEST(Ours)
AG News	81.66	84.63	**86.22**
DBpedia	95.40	**97.21**	97.16
IMDB	75.40	78.83	**90.65**
Elec	76.69	91.76	**92.21**
SST-2	79.44	84.00	**89.56**
Average	81.72 (+0.00%)	87.29 (+5.57%)	**91.16** (+9.44%)

Table 4: Test accuracy (%) with only ten labels per class.

Experiments

Table 5: Performance comparison with non-BERT-based semi-supervised approaches. (Li and Ye 2018; Gururangan et al. 2019; Dai and Le 2015; Li and Sethy 2020) (RL: Reinforcement Learning, Adv.: Adversarial, Temp. Ens.: Temporal Ensemble, Layer Part.: Layer Partitioning)

Datasets	Model	# of labels	Acc.
IMDB	CEST (ours)	30	90.2
	Variational Pre-training	200	82.2
	RL + Adv. Training	100	82.1
	SeqSSL + Self-training	100	79.7
	Layer Part. + Temp. Ens.	100	75.8
	SeqSSL + Adv. Training	100	75.7
	Layer Part. + Π model	100	69.3
AG News	CEST (ours)	30	87.1
	Variational Pre-training	200	83.9
	SeqSSL + Self-training	100	78.5
	SeqSSL + Adv. Training	100	73.0
DBpedia	CEST (ours)	30	98.6
	RL + Adv. Training	100	98.5
	SeqSSL + Self-training	100	98.1
	SeqSSL + Adv. Training	100	96.1

Thanks!