A Bit Bayesian Facilitates Efficient Training in Token Classification

Anonymous ACL submission

Abstract

Token classification is a fundamental subject in computational linguistics. Token classification models, like other modern deep neural network models, are usually trained on the entire training set in each epoch, while research has found the entirety of the training data may not be needed in later epochs of training. Moreover, over-training on data that are properly handled may poison the model. Inspired by human pedagogy, we propose a teacher-aware learning structure for token classification models. After each epoch of training, the teacher selects data it is uncertain of and data it predicts differently from the student, which are passed into the structure for training in the next epoch. As a proof of concept, we use a Bayesian linear classifier as the teacher and two commonly used backbone models as the student. Experiments show our method reduces the number of training iterations and improves model performance in most cases.

1 Introduction

Token classification tasks, such as Named Entity Recognition (NER) and Part-Of-Speech Tagging (POS tagging), are essential to the study of linguistics and natural language processing. In most works, token classification models are trained on the entire training set, i.e. the entire training set is fed forward and backward through the network in each epoch. This procedure implies that *all data are equal*, while studies have shown otherwise, that some data are well handled in early phases of training and induce less shift in weights among later epochs (Loshchilov and Hutter, 2015; Katharopoulos and Fleuret, 2018). It has been shown that over-training on data that have a minimum training loss may penalize the model (Fan et al., 2017; Li et al., 2021). Thus, it is favorable to design an efficient training strategy capable of reducing training on properly handled data.

In this work, we propose a teacher-aware structure to facilitate efficient training in token classification tasks. In human pedagogy, teachers and students interact with each other: teachers adjust their teaching for different students and students provide feedback for their teachers. This dynamic cooperative process can also lead to a half-the-effort-twice-the-result outcome in machine learning (Matiisen et al., 2017; Yuan et al., 2021). In our structure, the teacher interacts with the student via *uncertainty sampling*: after each training epoch, the teacher goes through the ENTIRE training set and selects data that it is uncertain of[1] and data that the student predicts differently, which are passed into the structure for training in the next epoch. Our approach reduces the number of training iterations and features a dynamic data sampling process, i.e. the teacher selects more data to train on when it is not certain or there is a discrepancy between the predictions from the student. As the teacher becomes confident with more data among the training set with gaining agreement with student's predictions, the teacher tends to select fewer data to train later on.

The key contribution of this work is an efficient training strategy for token classification. As a proof of concept, we use a Bayesian linear classifier as the teacher. We use BERT and Bi-LSTM that are widely used in token classification tasks as the students, and test on NER and POS tagging. Experiments show that our structure is able to reduce the number of training iterations and improves model performance in most cases.

2 Related Works

State-of-the-art token classification models (Yamada et al., 2020; Schweter and Akbik, 2021; Wang et al., 2021) are trained on the entire training set in each epoch. Alternatively, studies have

[1] Unless otherwise specified, *uncertainty* refers to *predictive uncertainty* henceforth

shown that some data are less informative, which should be considered less frequently during training (Loshchilov and Hutter, 2015; Katharopoulos and Fleuret, 2018; Sinha et al., 2020). Moreover, continuing training on less informative data may affect the model's performance (Li et al., 2021). A straightforward strategy to address this problem is to aggregate the largest k training loss (Fan et al., 2017). In such a method, k is a fixed value when training proceeds; however, k should be *dynamic*, since the extended data that are handled by the model vary in different phases of training. Thus, we believe that a dynamic efficient training strategy is vital for token classification and related learning tasks.

Inspired by human cognition and pedagogy, teacher-aware learning is an important strategy in curriculum learning (Bengio et al., 2009). Matiisen et al. (2017) find that a teacher-student curriculum learning framework leads to faster learning in sampling sub-tasks from a complex task. Yuan et al. (2021) propose a teacher-aware learner based on gradient optimization that is capable of bringing global and local improvements.

In the structure we proposed, the teacher uses *uncertainty sampling* to dynamically sample data from the training set. *Uncertainty sampling* is an effective approach to acquire informative data in active learning (Settles, 2009; Yang et al., 2015), based on which we implement a slightly different strategy, where the consistency between the output of the teacher and student is also considered.

In our work, we are in favor of a teacher model that is as simple as possible. Inference through a Bayesian neural networks is the principled way to obtain uncertainty, and attempts are made to tackle the intractable nature of Bayesian neural networks. Blundell et al. (2015) introduce *Bayes by Backprop*, which learns a Bayesian neural network by minimizing the Kullback-Leibler (KL) divergence between a diagonal Gaussian distribution and the true posterior. We train a Bayesian linear classifier which takes the output of the penultimate layer of the student model as the input, which is inspired by the implementation of Last Layer Laplace Approximation, that a Gaussian approximation to the last layer of a ReLU network is sufficient for yielding calibrated uncertainty estimations (Kristiadi et al., 2020).

3 Method

We train a Bayesian linear classifier as the teacher model to illustrate our points. Given a sequence $x = [x_1, ..., x_n]$ and its tags $y = [y_1, ..., y_n]$ where n is the sequence length, it goes through the student model and weights $\mathbf{w}$ is updated by gradient descent (GD). The Bayesian classifier takes the output of the penultimate layer of the student model as the input, and it is trained via *Bayes by Backprop*. After each epoch, we use uncertainty sampling to sample data from the ENTIRE training set, and the selected data are passed into the structure for training in the next epoch.

3.1 Bayes by Backprop

BAYES BY BACKPROP learns a probability distribution on the weights of a neural network (Blundell et al., 2015). In our work, it finds the parameter $\theta = (\mu, \rho)$, defined by a mean μ and a standard deviation parameter ρ, that minimizes the Kullback-Leibler (KL) divergence between a diagonal Gaussian distribution $q(\mathbf{w}_c|\theta)$ and the true Bayesian posterior of the weights given the training data $P(\mathbf{w}_c|\mathcal{D})$, where $\mathbf{w}_c$ denotes the weights of a linear classifier:

$$\mathcal{F}(\mathcal{D}, \theta) = KL[q(\mathbf{w}_c|\theta)||P(\mathbf{w}_c|\mathcal{D})]. \quad (1)$$

The cost in Eq.1 is approximated using Monte Carlo sampling:

$$\mathcal{F}(\mathcal{D}, \theta) \approx KL[q(\mathbf{w}_c^{(i)}|\theta)||P(\mathbf{w}_c^{(i)}|\mathcal{D})] \quad (2)$$

where $\mathbf{w}_c^{(i)}$ is the i^{th} Monte Carlo sample drawn from $q(\mathbf{w}_c|\theta)$. The approximation in Eq.2 is minimized by optimizing the function as follows:

$$f(\mathbf{w}_c, \theta) = \log q(\mathbf{w}_c|\theta) - \log P(\mathbf{w}_c)P(\mathcal{D}|\mathbf{w}_c).$$

To update θ, a noise factor ϵ is sampled from $\mathcal{N}(0, I)$, and let $\mathbf{w}_c = \mu + \log(1 + \exp(\rho)) \circ \epsilon$, where $\circ$ is point-wise multiplication. The parameters of θ is updated by back-propagation. We follow Blundell et al. (2015), using a scale mixture of two Gaussians as the prior:

$$P(\mathbf{w}_c) = \prod_i \pi\mathcal{N}(\mathbf{w}_c^{(i)}|0, \sigma^2) + (1-\pi)\mathcal{N}(\mathbf{w}_c^{(i)}|0, \varphi^2)$$

where σ^2 and φ^2 are the variances of the component distributions and π is a probability.

3.2 Uncertainty sampling

UNCERTAINTY SAMPLING samples data using predictive uncertainty translated from the uncertainty in weights:

$$P(\bar{y}|x^*, \mathcal{D}) = \mathbb{E}_{P(\mathbf{w}_c|\mathcal{D})}\left[P(\bar{y}|x^*, \mathbf{w}_c)\right]$$

where x^* is the output of the penultimate layer of a student model given x, and $\bar{y}$ is a predicted tag. After each epoch of training, we select uncertain data and data for which the teacher and student model predict differently from the ENTIRE training set, and feed them into the teacher and student model for training in the next epoch. Specifically, given a student model $\mathcal{M}(\cdot)$, a Bayesian classifier teacher $\mathcal{B}(\cdot)$, a training set $\mathcal{D}$ (with length m), sample times n, and a frequency threshold t, we perform uncertainty sampling as follows:

Algorithm 1 Uncertainty Sampling

```
inputs M(·), B(·), D, n, t
for i = 1, 2, ..., m do
    x ← D[i], x* ← M*(x), r ← []
    for j = 1, 2, ..., n do
        ε ~ N(0, I)
        w_c ← μ + log(1 + exp(ρ)) ∘ ε
        r[j] ← B_{w_c}(x*)
    end for
    τ ← the frequency of the mode in r
    if B_μ(x*) ≠ M(x) or τ < t then
        yield x
    end if
end for
```

$\mathcal{M}^*(\cdot)$ denotes the first to penultimate layers of the student model. We sample $\mathbf{w}_c$ for n times, which gives us n predictions of the teacher (stored in r). We find the frequency of the mode of the predictions τ and use it as the uncertainty estimation. If the prediction of the teacher when $\mathbf{w}_c = \mu$ does not equal to the prediction of the student model or τ is less than a threshold t (a low τ indicates high uncertainty), x will be used for training in the next epoch.

4 Experiments

4.1 Experiment settings

We use BERT and BiLSTM as the student model and experiment on CoNLL2003 (Tjong Kim Sang, 2002; Sang and De Meulder, 2003) and Penn Treebank (Marcus et al., 1993). We use the same configuration for the Bayesian linear classifier teacher in all experiments. We set the sample times n to 5. We consider a strict uncertainty sampling strategy, where we set the threshold t to 1, i.e. the teacher must be absolutely certain to an input before it is passed for training in the next epoch.

We compare our approach to standard training, i.e. train the student model on the entire training set in each epoch. We experiment on Top-k Loss (Fan et al., 2017) to inspect our method's capability of reducing the punishment caused by continuing training on properly handled data. We use a mini-batch variant of Top-k Loss, where k^* indicates the proportion of samples picked from a batch of data. For instance, $k^* = 0.8$ means we sample 80% data with the highest loss from a batch, which are used for update. In our experiments, we pick up k^* values such that the total number of data whose loss is used for update is close to the number of iterations in teacher-aware training.

The rest of implementation details can be found in Appendix.

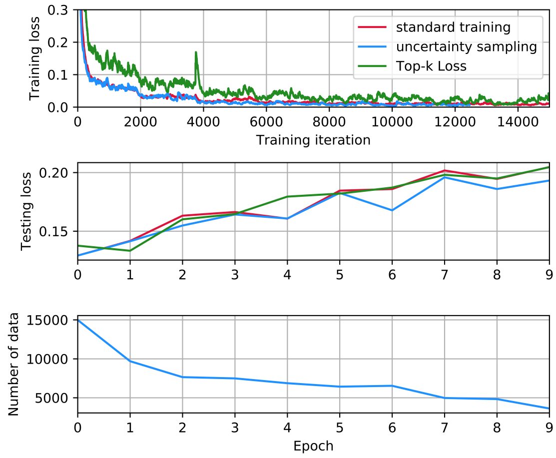

Figure 1: The training curve, testing loss, and number of training data in each epoch, trained with BERT on CoNLL2003 (EN) under different training methods. Results are averaged across 3 runs.

4.2 Results

Figure 1 shows that the model trained using our method has a slightly faster convergence rate and a lower testing loss compared to other methods. The teacher samples more data in early epochs than late epochs, which suggests that there is more discrepancy between the teacher and student, or the teacher is uncertain of most of the data when training begins; when training proceeds, the teacher and student become more equivocal and the teacher is certain to more data, resulting in less data being

DATASET	STUDENT	METHOD	ITERATIONS	TIME	F1%
CoNLL2003 (EN)	BERT	teacher-aware	9134±666	1276±64	**89.05±0.06**
		standard training	18740	1573±2	88.84±0.05
		Top-k ($k^* = 0.50$)	18740	-	88.37±0.08
	BiLSTM	teacher-aware	2492±252	307±18	76.70±1.20
		standard training	6600	390±0	**76.77±0.92**
		Top-k ($k^* = 0.40$)	6600	-	73.59±1.33
CoNLL2003 (ES)	BERT	teacher-aware	7648±81	954±6	88.38±0.11
		standard training	10400	878±1	88.28±0.09
		Top-k ($k^* = 0.75$)	10400	-	**88.53±0.64**
	BiLSTM	teacher-aware	2601±186	250±15	**75.90±2.74**
		standard training	3900	210±0	73.22±5.14
		Top-k ($k^* = 0.68$)	3900	-	73.26±2.66
CoNLL2003 (NL)	BERT	teacher-aware	10355±329	1402±34	**87.73±0.49**
		standard training	19760	1670±1	85.26±1.11
		Top-k ($k^* = 0.50$)	19760	-	86.30±1.18
	BiLSTM	teacher-aware	2533±674	338±42	**67.15±0.40**
		standard training	7410	420±0	66.13±0.35
		Top-k ($k^* = 0.35$)	7410	-	63.87±2.84
Penn Treebank	BERT	teacher-aware	36929±1003	4650±90	**93.27±0.39**
		standard training	49790	4219±6	92.56±1.35
		Top-k ($k^* = 0.75$)	49790	-	93.03±0.23
	BiLSTM	teacher-aware	18176±72	1846±4	**88.26±1.46**
		standard training	18690	1125±2	87.63±1.04
		Top-k ($k^* = 0.99$)	18690	-	87.72±1.89

Table 1: Number of training iterations in student model, training time (second), and test results (F1%) on CoNLL2003 and Penn Treebank under different student models and training methods. We do not report the training time of models trained using Top-k Loss, for it does not reduce training iterations. Results are averaged across 3 runs.

selected out for training. Our structure features a dynamic sampling process, distinct from methods such as Top-k Loss which selects a fixed number of data each time.

Table 1 displays the results of the experiments. We only report the number of training iterations in student model, since the training of the student model costs much more computational power than that of the teacher model. Our approach reduces the number of training iterations in all runs. In some runs, our method reduces more than 60% training iterations. Models trained using our approach outperform those using standard training in 6 out of 8 sets. We also observe a better performance in models trained using our approach than those using Top-k Loss. The training time of models trained using our approach is competitive, which is composed of 3 parts: the training time of the teacher model, the training time of the student model, and the sampling time.

5 Conclusion

We propose a teacher-aware structure that is able to facilitate efficient training in token classification. Such structure is simple yet capable of self-designing the training curriculum by taking advantage of uncertainty data sampling. We used two backbone models that are widely used in state-of-the-art token classification models as the student, and use a Bayesian linear classifier as the teacher. Among different experiments varied by tasks, source languages, and data-set size; our structure proved to be able to reduce the number of training iterations/training time, with no trade-off in the model's performance.

References

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning.

Charles Blundell, Julien Cornebise, Koray

Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural network. In *International Conference on Machine Learning*, pages 1613–1622. PMLR.

José Cañete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang, and Jorge Pérez. 2020. Spanish pre-trained bert model and evaluation data. In *PML4DC at ICLR 2020*.

Wietse de Vries, Andreas van Cranenburgh, Arianna Bisazza, Tommaso Caselli, Gertjan van Noord, and Malvina Nissim. 2019. BERTje: A Dutch BERT Model. arXiv:1912.09582.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Yanbo Fan, Siwei Lyu, Yiming Ying, and Bao-Gang Hu. 2017. Learning with average top-k loss.

Angelos Katharopoulos and François Fleuret. 2018. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pages 2525–2534. PMLR.

Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. 2020. Being bayesian, even just a bit, fixes overconfidence in relu networks. In *International Conference on Machine Learning*, pages 5436–5446. PMLR.

Zhiyuan Li, Tianhao Wang, and Sanjeev Arora. 2021. What happens after sgd reaches zero loss? –a mathematical framework.

Ilya Loshchilov and Frank Hutter. 2015. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330.

Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. 2017. Teacher-student curriculum learning.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Stefan Schweter and Alan Akbik. 2021. Flert: Document-level features for named entity recognition.

Burr Settles. 2009. Active learning literature survey.

Samarth Sinha, Zhengli Zhao, Anirudh Goyal, Colin Raffel, and Augustus Odena. 2020. Top-k training of gans: Improving gan performance by throwing away bad samples.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*.

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, and Kewei Tu. 2021. Automated concatenation of embeddings for structured prediction.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. Luke: Deep contextualized entity representations with entity-aware self-attention.

Yi Yang, Zhigang Ma, Feiping Nie, Xiaojun Chang, and Alexander G Hauptmann. 2015. Multi-class active learning by uncertainty sampling with diversity maximization. *International Journal of Computer Vision*, 113(2):113–127.

Luyao Yuan, Dongruo Zhou, Junhong Shen, Jingdong Gao, Jeffrey L. Chen, Quanquan Gu, Ying Nian Wu, and Song-Chun Zhu. 2021. Iterative teacher-aware learning.

A Implementation details

For experiments with BERT, we use bert-base-cased (Devlin et al., 2018) on CoNLL2003 (EN) and Penn Treebank, dccuchile/bert-base-spanish-wwm-cased (Cañete et al., 2020) on CoNLL2003 (ES), and GroNLP/bert-base-dutch-cased (de Vries et al., 2019) on CoNLL2003 (NL) as the encoder. The implementation is based on Hugging-face Transformers (Wolf et al., 2020). The model is optimized using Adam with a learning rate of 4e-5. We set batch size to 8 and train the model 10 epochs. The model contains 108M parameters.

For experiments with BiLSTM, we produce default word-level embeddings with size 768 and use 2 BiLSTM layers with hidden size 768. We add a dropout layer before the last BiLSTM layer and the linear classifier, with a rate of 0.2. The model is optimized using Adam with a learning rate of 4e-3, and we train the model 30 epochs with a batch size of 64. The model contains 28M parameters.

We do not alter the configuration of the Bayesian linear classifier teacher in one and another experiment. It is optimized using Adam with a learning rate of 4e-5. The Bayesian linear classifier contains 14K parameters. For each piece of training data, it samples 2 times to update θ. We use a strict uncertainty sampling strategy, where we set the threshold t to 1.

All of the experiments are run on Nvidia GeForce RTX 3090.