Delta-training: Simple Semi-Supervised Text Classification using Pretrained Word Embeddings

Hwiyeol Jo
Seoul National University
hwiyeolj@gmail.com

Ceyda Cinarel
Seoul National University
snu.ceyda@gmail.com

Abstract

We propose a novel and simple method for semi-supervised text classification. The method stems from the hypothesis that a classifier with pretrained word embeddings always outperforms the same classifier with randomly initialized word embeddings, as empirically observed in NLP tasks. Our method first builds two sets of classifiers as a form of model ensemble, and then initializes their word embeddings differently: one using random, the other using pretrained word embeddings. We focus on different predictions between the two classifiers on unlabeled data while following the self-training framework. We also use early-stopping in meta-epoch to improve the performance of our method. Our method, Delta-training, outperforms the self-training and the co-training framework in 4 different text classification datasets, showing robustness against error accumulation.

1 Introduction

1.1 Motivation

Classifiers using deep learning algorithms have performed well in various NLP tasks, but the performance is not always satisfactory when utilizing small data. It is necessary to collect more data for acquiring better performance. Although collecting unlabeled text data is relatively easy, labeling in and of itself requires a considerable amount of human labor. In order to incorporate unlabeled data into a task, we have to label the data in accordance to class policies of the task, but the labeling process requires not only human labor but also domain knowledge on the classes.

Semi-supervised learning (Li and Liu, 2003; Zhu, 2006; Chapelle et al., 2009) can be considered a potential solution that utilizes both labeled data and unlabeled data when building a classifier. The simplest form of semi-supervised learning is self-training (Yarowsky, 1995), which first builds a classifier using labeled data, and then label the unlabeled data. After which the most confident label prediction is added to training set and the process is repeated. The unlabeled data can help address data sparsity, but classification errors might be accumulated along the process.

We combine self-training with the hypothesis that a classifier with pretrained word embeddings (m_{emb}) is always better than a classifier with randomly initialized word embeddings (m_{rand}), as empirically observed in various NLP tasks (Turian et al., 2010). Our method follows the self-training framework but rather focuses on the different predictions of two sets of classifiers on unlabeled data. Therefore we can filter out incorrectly predicted data and correctly predicted data by both classifiers, which are less informative to the classifiers. On the other hand, differently predicted data are much more informative since much of the performance gap between the classifiers come from the different predictions. Although the differently predicted data may introduce some noise like correctly predicted by m_{rand} but incorrectly predicted by m_{emb}, we believe that the noise is relatively small when compared with benefits.

1.2 Contributions

Our contributions in this paper can be summarized as follows:

- We propose a variation of self-training framework: Delta(Δ)-training, which harnesses differently predicted labels between two sets of classifiers.
- Along with early-stopping in iterative training process, our framework outperforms the conventional self-training and co-training framework.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 3458–3463,
Hong Kong, China, November 3–7, 2019.

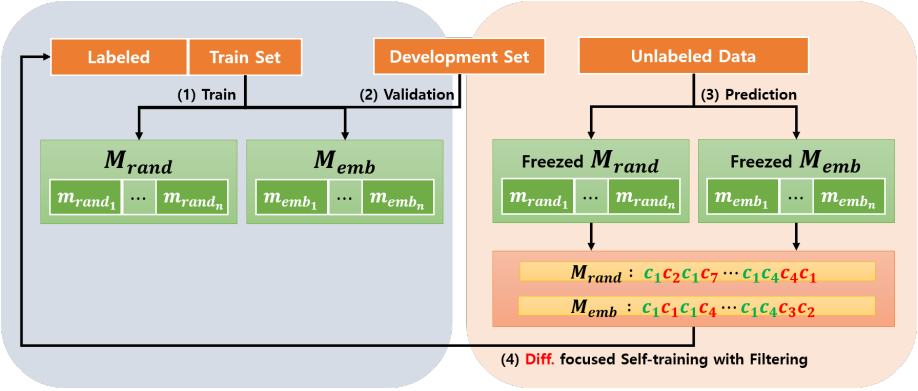

Figure 1: The flow of Δ-training framework. M_{rand} and M_{emb} are ensembled classifiers using randomly initialized word embeddings and pretrained word embeddings, respectively. (1) We first train the sets of classifiers using training set, (2) do early-stopping using development set, (3) predict the labels of unlabeled data using the sets of classifiers trained at (1), and (4) select the high confidence labels differently predicted by each set of classifiers, adding them to training set. While following the framework, we do early-stopping in meta-epoch with the development set.

2 Preliminary

Self-training. Given labeled data $\{(x_1, y_1), \cdots, (x_n, y_n)\}$ and unlabeled data $\{(x_{n+1}), \cdots, (x_{n+l})\}$, self-training (Yarowsky, 1995) first builds a model m using labeled data. Next, it simply predicts the unlabeled data using pretrained model m. If the confidence score of the predicted label is higher than a predefined threshold T, then adds the label-by-prediction data to the training set. This simple approach has generated variations such as calibration (Guo et al., 2017), and online learning (Abney, 2007).

Pretrained Word Embeddings. Pretrained word embeddings are based on the distributed representation hypothesis that a word can be represented as an n-dimensional vector (Mikolov et al., 2013). Most of the algorithms are based on the basic idea of CBoW and skip-gram. Both algorithms learn word vectors by maximizing the probability of occurrence of a center word given neighbor words or neighbor words given a center word. With this unsupervised approach, we can represent semantic and relational information. The pretrained word vectors from very large corpus are used to initialize word vectors for classifiers, performing better than randomly initialized word vectors (Turian et al., 2010).

Model Ensemble. Model ensemble (Opitz and Maclin, 1999) is using a combination of models to increase accuracy and get confidence scores on predictions. There are two types of ensemble methods, bagging and boosting. Bagging averages the predictions over a collection of classifiers whereas boosting weights the vote with a collection of classifiers.

3 Proposed Method: Δ-training

The overall process of our framework is illustrated in Figure 1.

3.1 Different Prediction focused Self-training

Our method consists of two classifiers: one is randomly initialized (m_{rand}; random network), and the other is using pretrained word vectors (m_{emb}; embedded network). When ensembling, we duplicate the same classifier, $M_{rand} = (m_{rand_1}, \cdots, m_{rand_n})$ and $M_{emb} = (m_{emb_1}, \cdots, m_{emb_n})$, respectively.

We adopt bagging to increase the cases that (1) both m_{rand} and m_{emb} predict the labels of data correctly, and (2) m_{rand} predicts incorrectly but m_{emb} predicts correctly. Model ensemble is also used to pick out label-by-prediction data with high confidence, which will be used for self-training. Intuitively, the benefits of Δ-training are maximized when the performance gap between two sets of classifiers is large. Also, we can ensure the performance gap not only by using pretrained embeddings but also through the ensemble setting.

First, we train the classifiers using the training set with early-stopping, and return their predictions on unlabeled data. We consider the predictions of M_{emb} on the unlabeled data as label-by-prediction since M_{emb} always outperforms M_{rand} according

to our hypothesis. The hypothesis will be confirmed in Section 7.

After labeling the unlabeled data, we select the data with conditions that (a) each ensembled classifiers are predicting the same class, and (b) the predictions of M_{rand} and M_{emb} are different. Condition (a) helps to pick out the data labeled with high confidence by the classifiers and Condition (b) helps to pick out the data which is incorrect in M_{rand} but correct in M_{emb}. The ratio in which labels might be correct in M_{rand} but incorrect in M_{emb} is relatively small than vice versa (will be also presented in Section 7). We add the selected data and its pseudo-label by M_{emb} to training set, and then train the classifiers again from the very first step to validate our hypothesis. If we do not start from the very first step, it might cause M_{emb} to overfit and perform worse than M_{rand}.

We denote one such iterative process, training and pseudo-labeling, as a *meta-epoch*.

3.2 Early-Stopping in Meta-Epoch

Using the development set in every meta-epoch, we do early-stopping during the different prediction focused self-training. As M_{rand} keeps learning based on the predictions of M_{emb}, the size of data which is incorrectly predicted by M_{rand} but correctly predicted by M_{emb} will decrease. Likewise, the size of data which is correctly predicted by both M_{rand} and M_{emb} will increase. Therefore, after early-stopping in meta-epoch, we simply add all the unlabeled data with its pseudo-labels to the training set. In this way, we can fully benefit from different prediction focused self-training and save training time.

4 Experiment Data

We use **GloVe** (Pennington et al., 2014) `glove.42B.300d` as word embedding for M_{emb}. We also perform word vector post-processing method, **extrofitting** (Jo, 2018), to improve the effect of initialization with pretrained word embeddings on text classification, as described in their paper.

We use 4 text classification datasets; **IMDB review** (Maas et al., 2011), **AGNews**, **Yelp review** (Zhang et al., 2015), and **Yahoo!Answers** (Chang et al., 2008).

	IMDB Review	**AGNews**	**Yelp Review**	**Yahoo Answer**
Train	212	1,020	5,525	1,136
Test	25,000	7,600	50,000	23,595
Dev	38	180	975	201
Unlabed	24,750	118,800	643,500	132,366
#Class	2	4	5	17

Table 1: The data split information of text classification datasets. We use 1% of the training data and remove the labels of the remaining training data, using them as unlabeled data.

5 Experiment

5.1 Data Preparation

To emulate the environment with only a few labeled training examples, we take only 1% of the original training set and remove the label of the remaining training set, which will be referred to as unlabeled data. Next, we assign 15% of the training set to the development set. The development set is used to determine early-stopping for every epoch and meta early-stopping for every meta-epoch. The split data size is presented in Table 1. IMDB reviews dataset has own unlabeled data but we do not use them in order to track and report on the difference between predicted labels and true labels.

5.2 Classifier

We select TextCNN (Kim, 2014) as our classifier. Due to the simple but high performance nature of TextCNN, the model can represent deep learning classifiers, and is easy to ensemble as well. We use the first 100 words of data in 300 dimensional embedding space. The model consists of 2 convolutional layers with the 32 channels and 16 channels, respectively. We adopt multiple sizes of kernels–2, 3, 4, and 5, followed by ReLU activation (Hahnloser et al., 2000) and max-pooled. We concatenate them after every max-pooling layer. We train the model using Adam optimizer (Kingma and Ba, 2014) with 1e-3 learning rate.

5.3 Ensemble Settings

In the experiment, the number of embedded model ensemble (M_{emb}) is 3 and we do not ensemble random model ($M_{rand} = m_{rand}$) for simplicity. The baselines also use the same ensemble settings for fair comparison.

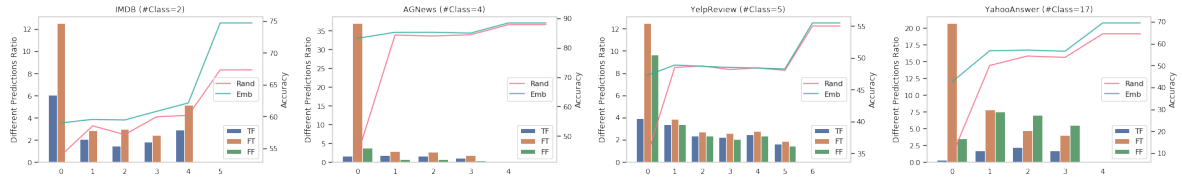

Figure 2: The training curve and the ratio of differently predicted label during Δ-training. The x-axis indicates a training process (meta-epoch). TF, FT, and FF denote that correctly predicted by M_{rand} but incorrectly predicted by M_{emb}, incorrectly predicted by M_{rand} but correctly predicted by M_{emb}, and incorrectly predicted by both M_{rand} and M_{emb}, respectively. We extend the end of performance lines to indicate final accuracy.

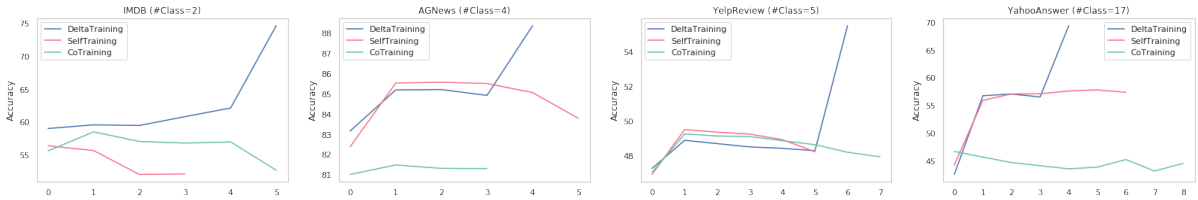

Figure 3: The performance of Δ-training compared with self-training and co-training. Our method largely improves the performance in binary classification but slightly show degraded performance prior to early-stopping in meta-epoch. Then, Δ-training finally brings significant performance gain after meta-level early stopping, training on all the remaining pseudo-labeled data.

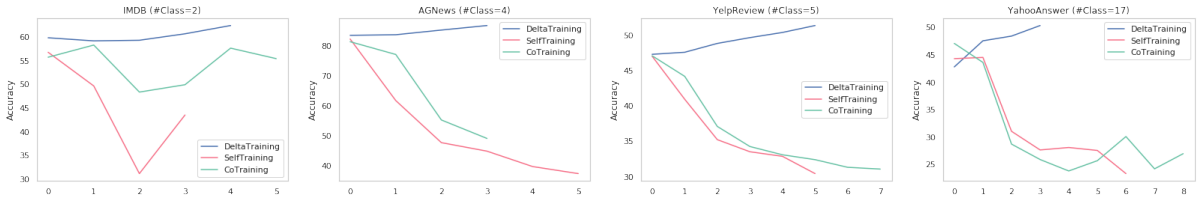

Figure 4: The performance of Δ-training and other frameworks on unlabeled data. We recover the removed true labels and track the model performance. Δ-training is robust against error accumulation.

6 Related Works

Δ-training is closely related to **Self-training** (Yarowsky, 1995), and **Tri-training with Disagreement** (Søgaard, 2010). Tri-training (Zhu, 2006) uses 3 classifiers to vote their classification results and labels them if all the classifiers agree with the prediction. Its extension, Tri-training with Disagreement, also uses 3 classifiers but the method utilizes a disagreement that pseudo-labels on unlabeled data if two classifiers agree with the labels but one classifier disagrees with the labels. The differences with our method respectively are (1) we harness different predictions of classifiers, and (2) we use a single model architecture where word embeddings are initialized differently.

The existing semi-supervised solutions using 2 classifiers such as **Co-training** (Blum and Mitchell, 1998) cannot be fully compared with ours for (2) that a single architecture should be used. The method is built on 2 different classifiers as having different views on data, and harnesses one's pseudo-labels to train the other classifier. Instead, we imitate the co-training as if M_{rand} and M_{emb} have different views on the data.

Refer to Ruder and Plank's work (2018) for further knowledge on those related works.

7 Result

The training curve and the ratio of differently predicted labels are presented in Figure 2. The training curves at x=0, at which point Δ-training is not applied yet, confirms our hypothesis–a classifier with pretrained word embeddings always outperforms the same classifier with randomly initialized word embeddings. Also, the ratio in which labels are correct in M_{rand} but incorrect in M_{emb} is relatively small (TF<FT) than vice versa. The ratio in which labels are incorrect both in M_{rand} and M_{emb} (FF) changes according to baseline accuracy and the number of classes.

The performance of Δ-training compared with

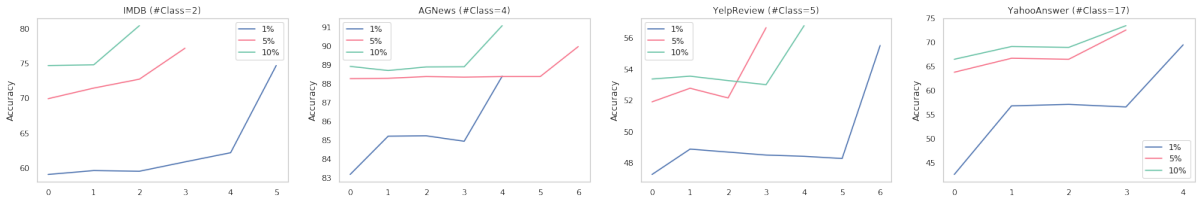

Figure 5: The performance of Δ-training with respect to initial training data size. Δ-training performs well in different training data size and is more useful when the training data is scarce.

self-training and co-training is presented in Figure 3. Our method largely outperforms the conventional self-training and co-training framework in binary class classification. In multi-class classification, picking different predictions is less effective because the data could be incorrectly predicted by both M_{emb} and M_{rand}. Therefore, after the early-stopping in meta-epoch, we simply add all the unlabeled data with its pseudo-labels to training set, which finally brings significant performance gain. In Figure 4, we observe that the performance of self-training and co-training decreases in unlabeled data after a few meta-epochs because of accumulated classification errors. On the other hand, our method is robust against error accumulation. As a result, the process of adding all the unlabeled data with its pseudo-labels to training set starts from enhanced and robust models.

We also report the effect of initial training data size in Figure 5. The result shows that Δ-training is more useful when the training data is scarce and also Δ-training works well even when there is relatively more data.

8 Conclusion

In this paper, we propose a novel and simple approach for semi-supervised text classification. The method follows the conventional self-training framework, but focusing on different predictions between two sets of classifiers. Further, along with early-stopping in training processes and simply adding all the unlabeled data with its pseudo-labels to training set, we can largely improve the model performance. Our framework, Δ-training, outperforms the conventional self-training and co-training framework in text classification tasks, showing robust performance against error accumulation.

Acknowledgments

The authors would like to thank Sang-Woo Lee, Woo-Young Kang, Dae-Soo Kim, and Byoung-Tak Zhang for helpful comments. Also, we greatly appreciate the reviewers for critical comments.

This work was partly supported by the Korea government (2015-0-00310-SW.StarLab, 2017-0-01772-VTT, 2018-0-00622-RMI, 2019-0-01367-BabyMind, 10060086-RISF, P0006720-GENKO), and the ICT at Seoul National University.

References

Steven Abney. 2007. *Semisupervised learning for computational linguistics*. Chapman and Hall/CRC.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

Ming-Wei Chang, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar. 2008. Importance of semantic representation: Dataless classification. In *AAAI*, volume 2, pages 830–835.

Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. 2009. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330.

Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas, and H Sebastian Seung. 2000. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947.

Hwiyeol Jo. 2018. Expansional retrofitting for word vector enrichment. *arXiv preprint arXiv:1808.07337*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Xiaoli Li and Bing Liu. 2003. Learning to classify texts using positive and unlabeled data. In *IJCAI*, volume 3, pages 587–592.

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

David Opitz and Richard Maclin. 1999. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Sebastian Ruder and Barbara Plank. 2018. Strong baselines for neural semi-supervised learning under domain shift. *arXiv preprint arXiv:1804.09530*.

Anders Søgaard. 2010. Simple semi-supervised training of part-of-speech taggers. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 205–208. Association for Computational Linguistics.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196. Association for Computational Linguistics.

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.

Xiaojin Zhu. 2006. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2(3):4.