Text Embedding Augmentation Based on Retraining With Pseudo-Labeled Adversarial Embedding

MYEONGSUP KIM AND PILSUNG KANG
School of Industrial & Management Engineering, Korea University, Seoul 02841, Republic of Korea
Corresponding author: Pilsung Kang (pilsung_kang@korea.ac.kr)

This work was supported by the Korea Institute for Advancement of Technology (KIAT) funded by the Korean Government (MOTIE), The Competency Development Program for Industry Specialist, under Grant P0008691.

ABSTRACT Pre-trained language models (LMs) have been shown to achieve outstanding performance in various natural language processing tasks; however, these models have a significantly large number of parameters to handle large-scale text corpora during the pre-training process, and thus, they entail the risk of overfitting when fine-tuning for small task-oriented datasets is conducted. In this paper, we propose a text embedding augmentation method to prevent such overfitting. The proposed method applies augmentation to a text embedding by generating an adversarial embedding, which is not identical to original input embedding but maintaining the characteristics of the original input embedding, using PGD-based adversarial training for input text data. A pseudo-label that is identical to the label of the input text is then assigned to adversarial embedding to conduct retraining by using adversarial embedding and pseudo-label as input embedding and label pair for a separate LM. Experimental results on several text classification benchmark datasets demonstrated that the proposed method effectively prevented overfitting, which commonly occurs when adjusting a large-scale pre-trained LM to a specific task.

INDEX TERMS Text embedding augmentation, adversarial training, pseudo-label, generating, retraining.

I. INTRODUCTION

Since the introduction of the transformer [1], several language models (LM) that employ the overall or partial structure of the transformer have been proposed for natural language processing (NLP) tasks, and they have shown remarkable performance compared with other conventional models. A well-known transformer-based LM, BERT [2], achieved the state-of-the-art performance in 11 NLP tasks at the time of its publication using a two-stage transfer learning framework, wherein unsupervised pre-training was primarily performed for large-scale unlabeled text corpora, and followed by fine-tuning to additionally update the entire model for a labeled specific target task dataset. After the success of BERT, various LMs employing the pre-training and fine-tuning learning framework based on the transformer architecture have been proposed and have achieved notable performances in a variety of NLP tasks such as text classification, question answering, and text summarization [3]–[8].

During pre-training, which is the first step of the two-stage transfer learning, an LM captures syntactic and semantic information through the interactions between tokens in the sequence and obtains generalized and contextualized representations that can be used for various NLP tasks [9], [10]. In order to obtain such general language representation, large-scale text corpora are generally used for pre-training, and an LM has a significantly large number of parameters to accommodate such data, resulting in high model complexity [11]. During fine-tuning, which is the second step, an LM is adapted to the labeled task dataset and obtains competency to perform a specific target task. However, labeled data resources are limited in many downstream tasks because it requires an extensive amount of time and cost for obtaining these data. Consequently, there is a risk of overfitting during fine-tuning probably caused by the high complexity of an LM and limited labeled task-specific data resources [11].
Overfitting refers to the situation that the model can only be well fitted to the training dataset, but it reports much lower performance for unseen data [12]. One of the primary causes of overfitting is that the size of the training dataset is insufficiently small with respect to the model complexity. With growing opportunities to learn noises included in the training data in such a situation, the model learns both the patterns found in the training data as well as uncommon noisy patterns, and therefore, the model’s error for an unseen data containing different noises from the training dataset becomes more prominent [12].

Regularization is one of the methods for preventing the overfitting of a deep learning model [12]. Regularization can be defined as “any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error.” [13]. Popular regularization methods include dropout [14], normalization [15], and data augmentation [16]. Dropout is a regularization method where multiple networks are learned as an ensemble by randomly deleting certain nodes in the base neural network [13], whereas normalization is a method applying estimated normalization statistics to the input of a hidden layer neuron [15].

Regularization techniques such as dropout and normalization are actively used in the NLP domain, whereas it is difficult to directly apply data augmentation in the NLP domain, because most existing data augmentation methods were originally developed for vision domain tasks [17]. For example, flipping, which reverses an image; cropping, which removes parts of an image; and rotation, which rotates an image in the training dataset, do not change semantic elements of the original image and preserve the label except for special cases [18]. When applying augmentation in the NLP domain, however, it is difficult to directly apply the previously mentioned methods to natural language tokens. For text data, even fine changes made to the tokens constituting the text can significantly change the meaning implied by the text [17], because tokens having similar forms may have completely different meanings (e.g., bad and bed) and the meaning of a single token can be vary depending on the context [10]. Therefore, transforming tokens of the text data may change the meaning of the entire text and even change the label; hence, augmentation in the NLP domain requires a more careful approach than augmentation in the vision domain [19].

Text augmentation methods, such as easy data augmentation (EDA) [20] and back-translation [21], have been proposed to address the problem that the augmentation technique used in the vision domain is difficult to be applied to the NLP domain. These methods improved the performance of existing models in various NLP tasks by replacing, deleting, or inserting natural language tokens in the text or translating the text using a neural machine translation model into another language and then back-translating into the original language. However, the two methods both entail the risk of the aforementioned problems as transformation is applied to the original text data at the token level. EDA has been found to change the semantic elements of the original input text [22], whereas back-translation has been reported to generate inaccurate text depending on the performance of the translator [17] or the vocabulary of the augmented text is reduced as specific tokens are repeated [23].

As another way to improve the generalization performance of an NLP model, adversarial training algorithms which add adversarial perturbations to word embeddings have been proposed [11], [24], [25]. Instead of applying transformation at the token level, these algorithms generated adversarial examples at the embedding level and used them to update the parameters of RNN-based models or transformer-based models. These adversarial training algorithms have been experimentally proven to effectively enhance the generalization performance of an LM, but they have a limitation in that the adversarial examples are not fully utilized because they are temporarily generated through random initialization and used only once for updating the LM parameters. Furthermore, these algorithms used the generated adversarial example as the intermediate output of regularization rather than as the data that can be used as the input of a separate model.

In this paper, we propose a text embedding augmentation method based on adversarial training. Following Zhu et al. [25], we first apply projected gradient descent (PGD) [26] based adversarial training to the embedding of an LM. Adversarial training is performed where a perturbation is trained to increase the loss of an LM within a space limited by the norm of a certain size, while parameters are trained to reduce the loss of an LM for an adversarial example in which a perturbation is added to the original embedding. In this process, the adversarial example generated in the form of the text embedding is saved and is considered to have the same label as the label of the original text data for performing training for a separate LM. Since the adversarial examples are generated by the size of the original data for every one epoch of adversarial training, the labeled embeddings can be augmented as adversarial training continues. The main contributions of this paper can be summarized as follows:

- We proposed a text augmentation method for embeddings rather than natural language tokens. Very fine changes in natural language tokens may lead to significant semantic changes, thus losing syntactic and semantic characteristics of the original text; however, by manipulating the embeddings within the limited space, it is possible to augment text embeddings while maintaining the characteristics of the original text data.
- The generalization performance of an LM can be improved by generating augmented embeddings that cannot be easily classified by the original LM. In the process of adversarial training, adversarial examples are generated to increase the loss of a model, and therefore, adversarial examples induce model misclassification. By assigning a pseudo-label that corresponds the label of original text data to an adversarial example and training them as the input and label pair for a separate model, the generalization performance of the LM can be improved.
The rest of this paper is organized as follows. In Section II, we briefly review the related work on text data augmentation and adversarial training for NLP. In Section III, we demonstrate our proposed method focusing on PGD-based adversarial training for LM and retraining with pseudo-labeled adversarial embeddings. In Section IV, experimental settings including datasets are demonstrated, followed by the analysis and discussion on the experimental results. Finally, in Section V, we conclude our work with some future research directions.

II. RELATED WORKS

A. TEXT DATA AUGMENTATION

Data augmentation has been widely used in the vision domain because it can improve the generalization ability of deep learning models by providing a sufficiently large amount of appropriately generated artificial training examples to avoid overfitting to an insufficient original dataset [18]. Despite its high usability in the vision domain, data augmentation has not been very useful in the NLP domain. This is mainly because contrast to pixels, the basic units of image, tokens, the basic units of NLP, are neither spatially and locally correlated nor position invariant. Therefore, it is difficult to directly apply various augmentation techniques that do not change the original label, such as flipping, cropping, and rotation, to tokens in NLP [17].

In light of this, various text data augmentation methods that are applicable to the NLP domain have been proposed. Wei and Zou [20] proposed an EDA method that can be applied to the text data where changes are made at the token level in which a specific word is replaced with a synonym, a random word is inserted, the locations of two words in a sentence are arbitrarily changed, or a random word is deleted. However, when EDA is applied to text data, it was found that the meaning of the original text data often changes [22]. To make matters worse, even the label for a specific task is changed in some cases after EDA was applied [19]. There have been cases where the sentence became illogical because the word order of the original text was changed [23].

Sennrich et al. [21] suggested a back-translation method wherein text data are augmented by translating the text data into another language using a neural machine translation model, followed by back-translation it into the original language. Back-translation has contributed to the improvement in the performance of multiple models in various NLP tasks, such as text classification, machine translation, and question answering [27]–[30]. However, back-translation also entails some problems. First, the vocabulary of the augmented text is reduced as a specific token is repeated in the text depending on the sampling method used in the language generation process [23]. Second, unnatural sentences that are uncommonly used may be generated [31]. In such a case, the probability of the generated text appearing in unseen data is decreased, which limits the possibility of significantly improving the generalization performance.

The proposed method differs from that in prior studies in that it performs text data augmentation while maintaining semantic elements of the original text data by manipulating text embeddings instead of manipulating tokens.

B. EMBEDDING LEVEL TEXT DATA AUGMENTATION

As another way of text data augmentation, several studies have been proposed to augment text data at the embedding level rather than at the token level [32], [33]. These methods first convert the discrete one-hot encoding of natural language tokens into embeddings in a lower-dimensional continuous space and then manipulate the embeddings in the continuous space to generate embeddings with characteristics similar to those of the original embedding. In general, it is known that embeddings derived from texts with similar meanings are located at a close distance in the embedding space [34], and an analogous relationship is established by addition or subtraction operations between embedding vectors [35]. As such, it has been reported that semantic information of text is expressed in the representation space [10]. Therefore, generated embeddings located at a close distance in the original embedding in the embedding space can be utilized as data augmentation for text corresponding to the embedding [36].

Papadiki [32] proposed methods for generating new embeddings based on the manipulations of embeddings using Gaussian noise, interpolation, and extrapolation. The method applying Gaussian noise was performed by adding the noise derived from the Gaussian distribution to the original embedding. In the methods employing interpolation and extrapolation, after obtaining the centroid of the three nearest embeddings to a specific embedding in the embedding space, the interpolation or extrapolation of the specific embedding and the centroid was used as the new embedding. In this method, specific hyperparameters were used for interpolation and extrapolation, and embeddings similar to the original embedding were generated by controlling the degree of interpolation or extrapolation using hyperparameters.

Guo et al. [33] proposed a method for augmenting word embeddings or sentence embeddings based on Mixup [37]. For two sentences, text embedding augmentation is performed by applying interpolation on a pair of word embeddings generated from each word constituting the sentence and a pair of labels corresponding to the sentence in the word embedding case. In the sentence embedding case, augmentation is performed by applying interpolation on a pair of sentence embeddings generated by applying CNN [38] or LSTM [39] to each word embedding from the sentence and a pair of labels for two sentences.

These methods contribute to the augmentation of text data through manipulation in the continuous embedding space. However, because no spatial constraints are imposed on the augmented embedding, there is a potential risk that embeddings that exceed the semantic boundary of specific embeddings may be generated in the process of applying noise, interpolation, and extrapolation. In addition, there is a limitation in that the embeddings are created arbitrarily without
directly using the loss information of a model in the process of creating new embeddings.

The proposed method differs from that in prior studies in that it sets a very small norm boundary around the original embedding and generates augmented embeddings within that boundary, and the loss of an LM is employed in the process of generating augmented embeddings.

C. ADVERSARIAL TRAINING FOR NATURAL LANGUAGE PROCESSING

Adversarial training is one of the techniques used for improving the robustness of a deep learning model to adversarial examples [25]. An adversarial example refers to an example causing misclassification of a model by applying minute changes to the original example [40]. For instance, a human determines the adversarial example in which a perturbation is added to image data as the same class as the original, but a model determines such an example as a different class from the original. For preventing the robustness of a model to adversarial examples from decreasing, adversarial training proceeds as follows to ensure the class of the adversarial example to be classified as the same class of the original example. First, the perturbation for the input is trained so as to increase the loss of the target model. Then, the trained perturbation is added to the original input to generate an adversarial example that causes misclassification of the model. Through the process of updating the model parameters to decrease the loss of the model using the generated adversarial example with the class information of the original example, the model gains the ability to classify the adversarial example as the class of the original input. In general, when adversarial training is performed in the vision domain, generalization performance is known to decrease, due to a trade-off between robustness and generalization performance [25]. In contrast, it is known that both generalization performance and robustness tend to increase when adversarial training is performed in the NLP domain [11, 24, 25, 41].

In the NLP domain, it is challenging to directly apply adversarial training of the vision domain, in which a gradient ascent is performed for the continuous input to increase the loss of the model, due to the discrete nature of the input token. Some adversarial training methods applicable in the NLP domain have been proposed to overcome such problems. Cheng et al. [41] proposed a method for generating an adversarial sentence that increases the loss of a neural machine translation model by replacing the token of the input sentence; however, the adversarial sentences thus generated were unnatural [31].

In recent years, adversarial training methods have been proposed where manipulations are made to embeddings of the token in a continuous feature space rather than to a natural language token itself. Zhu et al. [25] proposed a method generating an adversarial example for natural language embeddings by training a perturbation to increase the loss of a transformer-based LM based on PGD and performing adversarial training using such an example. Based on the method proposed by Zhu et al. [25], Jiang et al. [11] proposed a training method that gives smoothness to an LM through adversarial training to minimize the changes in the model output when a perturbation within a limited norm boundary is imposed on the input embedding. Their methods contributed to the significant improvement in the generalization performance of an LM by performing adversarial training for continuous natural language embeddings instead of discrete natural language tokens. However, the adversarial examples are temporarily generated through random initialization and used for training, and therefore, they can only be used once for the training of model parameters. In addition, these methods used adversarial examples as intermediate outputs belonging to a specific model and not as the data that could be used as an input to a separate model.

This study differs from previous studies in that adversarial training is performed for the embeddings of an LM and the label of the original text data is assigned to the generated adversarial example to be repeatedly used in a separate model.

III. THE PROPOSED METHOD

In this paper, we propose a text embedding augmentation method based on adversarial training for the embeddings of an LM. Our framework first generates an adversarial embedding by applying PGD-based adversarial training to the input sequence, and then assigns the target task label of the original input sequence to the generated sequence of adversarial embedding as a pseudo-label and uses them as the input for fine-tuning a separate pre-trained LM.

A. PGD-BASED ADVERSARIAL TRAINING FOR LANGUAGE MODEL

PGD-based adversarial training is used in this study for augmentation in which a perturbation generated within a norm boundary of a certain size is imposed on the input embedding in order to secure diversity for embeddings. Following Zhu et al. [25], PGD-based adversarial training for the sub-word token embedding among the embeddings of an LM is performed by detecting the optimal LM parameter \( \theta^* \) that satisfies the minimizing the maximum risk problem (1):

\[
\min_{\theta} \mathbb{E}_{(x, y) \sim \mathcal{D}} \left[ \max_{L \leq \epsilon} \mathbb{L}(f_{\theta}(x + \delta), y) \right], \tag{1}
\]

where \( Z = [z_1, z_2, \ldots, z_n] \) is a sub-word one-hot representation of the input sequence including special tokens such as [CLS] and [SEP], whereas \( X = VZ \) is sub-word embedding corresponding to the one-hot representation. The sub-word token refers to the token generated by the sub-word encoding method such as byte pair encoding (BPE) [42], whereas sub-word embedding refers to the embedding corresponding to the respective token. \( f_{\theta} \) represents the encoder of the LM, and \( \theta \) represents all trainable parameters in the LM. \( \mathcal{D} \) and \( y \) refer to the input distribution and label, respectively, while \( L \) refers to the loss function of an LM for a specific fine-tuning
FIGURE 1. Overall architecture. Overview of the proposed text embedding augmentation method based on retraining with pseudo-labeled adversarial embedding. The proposed method performs adversarial training using the input text data and task labels during the generating process. The sequence of the adversarial embedding for the input sequence is generated during adversarial training, and a pseudo-label that is identical to the label of original input is assigned to the sequence of adversarial embedding. Then, in the retraining process, the sequence of adversarial embedding and pseudo-label pairs are used as the input for fine-tuning a separate LM through batch resampling.

When performing adversarial training, a perturbation is trained so as to increase the loss of an LM when solving the inner maximization problem and added to the original input sub-word embedding. For preserving the semantic characteristics of the input embedding, the perturbation is trained within the norm boundary of limited size [25] in which the norm used in this process is represented as $\epsilon$. The input embedding to which the perturbation is added is referred to as an adversarial example; the generated adversarial example is located at a very close distance from the input embedding in the embedding space and has the same form as the input embedding, thus being referred to as the adversarial embedding in this paper. Adversarial embedding is generated through the multi-step update method shown in (2) below.

$$
\delta_{t+1} = \Pi_{||\delta|| \leq \epsilon} \left( \delta_t + \alpha \frac{g(\delta_t)}{||g(\delta_t)||} \right),
$$

(2)

where $g(\delta)$ defined as:

$$
g(\delta_t) = \nabla_{\delta} \mathcal{L}(f_\theta(X + \delta_t), y),
$$

$\alpha$ represents the update step size for training the perturbation, and $g(\delta)$ represents the gradient of the loss function with respect to the perturbation calculated in the $t$-th step. $\Pi$ performs projection onto the norm boundary when the perturbation goes out of the norm boundary during training.

Using this procedure, the perturbation is updated by the number of steps, which is a hyperparameter, within the norm boundary to increase the loss of an LM while solving the inner maximization problem.

The adversarial embedding generated while solving the inner maximization problem is reused as the input of the LM to solve the outer minimization problem so as to decrease the loss between the ground-truth label and model prediction. Consequently, the LM is trained to predict the adversarial embedding trained to increase the loss within the specific norm boundary as the same label as the original input embedding. Inner maximization problem and outer minimization problem constituting such PGD-based adversarial training are non-convex and non-concave, respectively; thus, there is a possibility of a saddle-point problem [25], but it has been proven that the overall process can be optimized stably using the stochastic gradient descent algorithm [26].

In general, when adversarial training is performed using image data as input, the generalization performance of a deep learning model is degraded [25], [43]. However, Zhu et al. [25] and Jiang et al. [11] reported that the generalization performance of an LM can be improved if adversarial training is performed using text embedding as input. However, in these methods, it is difficult to reproduce adversarial embeddings during the training process since the initial values of perturbations were set through random initialization, resulting in the generated adversarial embedding being used
overall process of the proposed method is shown in Figure 1. Generating pseudo-labeled adversarial embedding consists of two processes: constructing an augmented dataset by generating pseudo-labeled adversarial embedding for a specific LM (generating process), and then retraining a separate LM using the augmented dataset (retraining process). The overall process of the proposed method is shown in Figure 1.

\textbf{Algorithm 1 Generating Pseudo-Labeled Adversarial Embedding}

\textbf{Input:} $X = \{(Z, y)\}$: Original training dataset

\textbf{Require:} $\tau$: learning rate, $E_{\text{gen}}$: generation epochs, $K$: ascent steps, $\alpha$: ascent step size, $\epsilon$: norm boundary size

\begin{enumerate}
\item Augmented dataset $X_{\text{aug}}$
\item Initialize language model (generation) parameters $\theta_{\text{gen}}$
\item for generation epoch $e = 1, \ldots, E_{\text{gen}}$ do
\item Augmented batch $B_{\text{aug}}$
\item for minibatch $B : \{(Z_{\text{batch}}, y_{\text{batch}})\} \subset X$ do
\item Embedding $X = VZ_{\text{batch}}$
\item Initialize $\delta_0 \sim U(-\epsilon, \epsilon)$
\item for ascent step $t = 1, \ldots, K$ do
\item Update perturbation $\delta$ using gradient ascent
\item $g(\delta_{t-1}) = \nabla_{\delta} L(f_{\theta}(X + \delta_{t-1}, y_{\text{batch}}))$
\item $\delta_t = \Pi_{||\delta|| \leq (\delta_{t-1} + \alpha g(\delta_{t-1}) / ||g(\delta_{t-1})||})$
\item end for
\item Append adversarial embedding and pseudo-label
\item $B_{\text{aug}}.\text{append}((X + \delta, y_{\text{batch}}))$
\item Update language model parameters $\theta$ using gradient ascent
\item $\theta_{\text{gen}} = \theta_{\text{gen}} - \tau \nabla_{\theta} L(f_{\theta}(X + \delta), y_{\text{batch}})$
\item end for
\item Append augmented batch
\item $X_{\text{aug}}.\text{append}(B_{\text{aug}})$
\item end for
\end{enumerate}

\textbf{Algorithm 2 Retraining With Pseudo-Labeled Adversarial Embedding}

\textbf{Input:} $X_{\text{aug}}$: Augmented dataset

\textbf{Require:} $\tau$: learning rate, $E_{\text{gen}} = \text{len}(X_{\text{aug}})$: generation epochs, $E_{\text{ret}}$: training epochs for retraining model, $E_{\text{start}}$: start retraining index, $E_{\text{end}}$: end retraining index, where $E_{\text{start}} \leq E_{\text{end}} \leq E_{\text{gen}}$

\begin{enumerate}
\item Set part of augmented dataset as retraining dataset
\item Retraining dataset $X_{\text{ret}} = X_{\text{aug}}[E_{\text{start}} - 1 : E_{\text{end}}]$
\item Initialize language model (retraining) parameters $\theta_{\text{ret}}$
\item for retraining epoch $1, \ldots, E_{\text{ret}}$ do
\item Resample minibatch from retraining dataset
\item for minibatch $B : \{(X_{\text{ret}}, y_{\text{ret}})\} \subset X_{\text{ret}}$ do
\item Retrain with adversarial embedding
\item $\theta_{\text{ret}} = \theta_{\text{ret}} - \tau \nabla_{\theta} L(f_{\theta}(X_{\text{ret}}), y_{\text{ret}})$
\item end for
\item end for
\end{enumerate}

only once in the outer minimization problem for updating the parameters of an LM. In other words, in these methods, it is impossible to repeatedly use the adversarial embeddings for training as with the labeled data.

\section{B. Retraining With Pseudo-Labeled Adversarial Embedding}

In this section, we propose a text embedding augmentation method based on retraining with pseudo-labeled adversarial embedding. Our text embedding augmentation method consists of two processes: constructing an augmented dataset by generating pseudo-labeled adversarial embedding for a specific LM (generating process), and then retraining a separate LM using the augmented dataset (retraining process). The overall process of the proposed method is shown in Figure 1.

First, in the generating process, PGD-based adversarial training is performed for the embedding of a specific LM to generate pseudo-labeled adversarial embedding. In the process of solving the inner maximization problem, which is one of the procedures of PGD-based adversarial training, gradient ascent is applied to perturbation for sub-word token embedding formed from the input text data. In this paper, the embedding that corresponds to each token and becomes the training target of a perturbation is referred to as the target embedding. The perturbation updated by the number of ascent steps is added to the target embedding to generate an adversarial embedding. During this process, a very small norm boundary is formed around the target embedding, and the perturbation is trained within that boundary; thus, it is difficult for the perturbation to have a sufficient
FIGURE 2. Process of generating adversarial embedding. In the generating process of the proposed method, an adversarial embedding is generated using PGD-based adversarial training for the original embedding (a). Adversarial training is conducted so as to increase the loss in the (b) inner maximization process, thus generating an adversarial embedding to cause misclassification beyond the decision boundary. Then, the model parameters are trained so that the generated adversarial embedding can be correctly classified in the (c) outer minimization process. The adversarial embedding generated by repeating the above steps corresponds to the weak point of different model parameters as shown in (d), in which retraining the adversarial embedding helps strengthen these weak points.

FIGURE 3. Embedding space after retraining with pseudo-labeled adversarial embedding. A pseudo-labeled adversarial embedding in the proposed method is generated within a norm boundary of a limited size with respect to the input embedding, and thus is positioned close from the input embedding in the embedding space. (c) Repeatedly training the augmented pseudo-labeled adversarial embedding can expand the region of the input embedding, which can contribute to the generalization performance of an LM.

norm to change the target embedding into the embedding corresponding to another token within the embedding space. As a result, the generated adversarial embedding is positioned at a very close distance from the target embedding corresponding to each sub-word token constituting the input text data. Therefore, the adversarial embedding can maintain the characteristics of the target embedding in the embedding space fairly well and can be assumed to maintain the label of input text data [22]. Based on this assumption, the label for the adversarial embedding generated from the input embedding is assigned with the same label as the input text data, which is referred to as pseudo-label for adversarial embedding. Once the ascending step of the inner maximization problem is completed, the generated adversarial embedding and pseudo-label pairs are saved to form the augmented dataset. Then, the outer minimization problem is solved using adversarial embedding and pseudo-label pairs, which is identical to the procedure of PGD-based adversarial training explained in Section III-A. In the generating process, adversarial embedding and pseudo-label pairs are generated with the same number of sub-word token embedding and ground truth label pairs used for training after solving the inner maximization problem. Therefore, the original data are augmented by $E_{gen}$ times, which is the number of total epochs for which the generating process is performed. The overall procedure of the generating process is as shown in Algorithm 1.

Subsequently, the generated augmented dataset is used to train a subsequent pre-trained LM. The adversarial embedding and pseudo-label pairs included in the augmented dataset were involved in the training when solving the outer minimization problem of PGD-based adversarial training conducted to form the augmented dataset. In our proposed method, these adversarial embedding and pseudo-label pairs are saved and re-used to train a separate LM, which we call “retraining”. During the retraining process, the augmented dataset is entirely or partially used to fine-tune a separate pre-trained LM. Here, we constructed a minibatch used for retraining through random sampling on the entire or partially selected augmented dataset. As a result, the minibatch used in the retraining process includes adversarial embeddings generated using different LM parameters, thus securing additional diversity. The overall procedure of the retraining process is shown in Algorithm 2.
C. WHY ADVERSARIAL EMBEDDING?

An adversarial example trained for the input image in the vision domain has the same class as a benign image when determined by a human, but it induces misclassification of a model [40]. Image data having such characteristics are generally difficult to find in the real world; therefore, an adversarial example is highly likely to not appear in an unseen dataset [18]. Therefore, the generalization performance tends to decrease when adversarial training is performed [25], [43], which is deemed reasonable.

The adversarial embedding was generated through manipulation within the embedding space; hence, it is difficult to find a token in the real world that corresponds to each adversarial embedding. However, a transformer-based LM performs self-attention for the input embedding to reflect the information of other token embeddings included in the sequence, and the training process of forming contextualized representation that finely varies according to the token appearing concurrently is repeated [10]. Training the adversarial embedding generated by imposing a very small perturbation for maintaining the characteristics of the original sub-word embedding can contribute to the generalization of contextualized representation, which may be potentially generated for unseen data depending on the context of a sequence.

In addition, adversarial embeddings are generated using the perturbation trained so as to increase the loss of an LM in the process of solving the inner maximization problem of adversarial training. Because perturbation is trained within the norm boundary of a limited size, it can be assumed that the characteristics of sub-word embedding are maintained fairly well, without causing any semantic changes [25] but increasing the LM loss. Training such adversarial embeddings can contribute to strengthening the weak spot, which leads to misclassification of a model [18]. Adversarial embedding in this paper was generated so as to increase the loss of different LM parameters according to the generating process, as shown in Figure 2; consequently, it can strengthen the weak spot for different LM parameters through retraining. Moreover, as shown in Figure 3, repeating the retraining using adversarial embedding and pseudo-label pair can expand the region of a specific sub-word embedding within the embedding space, thus contributing to the improvement of generalization performance of the LM. In Section IV-F, the assumption proposed in this section and the expected effects are verified based on additional analysis results of the pseudo-label and the embedding space.

IV. EXPERIMENTS AND ANALYSIS

In this section, we verify the effectiveness of the proposed method on four natural language understanding datasets through comparative experiments with fine-tuning [2] and PGD-based adversarial training [25]. In addition, we verify the effectiveness of our augmentation method by applying it to a case with a small training dataset. Lastly, we verify the validity of assigning a pseudo-label to the adversarial embedding through an experimental analysis.

A. DATASETS

In this paper, we perform a comparative experiment of fine-tuning and PGD-based adversarial training on four natural language understanding datasets to evaluate the performance of our proposed augmentation method. The datasets used in the experiment are IMDB [44], Stanford Sentiment Treebank (SST) [45], Recognizing Textual Entailment (RTE) [46]–[49], and Question Natural Language Inference (QNLI) [50]. All datasets aim to perform text classification task, and accuracy was used as a performance evaluation metric. The target task, the number of classes in each dataset, and the number of training/validation/test instances are provided in Table 1.

- **IMDB**: IMDB review dataset consists of reviews collected from the IMDB website, allowing no more than 30 reviews for each movie. The dataset has the same number of positive and negative reviews and thus can be used for sentiment polarity classification. Positive reviews have a rating score of 7 or higher out of 10, whereas negative reviews have a rating score of 4 or lower out of 10 [44].

- **SST-2**: SST-2 dataset [45] consists of sentences extracted from movie reviews and emotional class labels for each sentence assigned by human annotators. The dataset used in this paper contains data having a label at the sentence level with a positive or negative class, and it aims to perform a sentiment polarity classification task [51].

- **RTE**: RTE dataset was generated using the dataset introduced in the PASCAL textual entailment challenge. The dataset used in this paper was proposed by Wang et al. [51] and generated by combining RTE1 [46], RTE2 [47], RTE3 [48], and RTE5 [49]. The task is to predict the entailment of two sentences in the dataset; each dataset having a two-class or three-class label was converted to have a two-class label of entailment or not-entailment [51].

- **QNLI**: The Stanford Question Answering Dataset (SQuAD) [50] consists of pairs of questions and paragraphs in which it is used in the question-answering task since one of the sentences constituting a paragraph contains the answer for the corresponding question. The dataset used in this study was generated by Wang et al. [51] who modified the SQuAD, and it consists of pairs of a sentence containing an answer and a corresponding question in a paragraph. The task using this dataset is to predict whether a given sentence contains the answer to a given question. The respective task belongs to the natural language inference (NLI) task [51].

B. EXPERIMENTAL DETAILS

The experiment was conducted by applying the proposed method to the pre-trained BERT [2]. The BERT provided by the Hugging Face [52] was employed as the pre-trained BERT, and the batch size for fine-tuning and generating, and retraining was 8 or 16.
TABLE 1. Dataset description. Task category, the number of classes, and the number of training (Train), development (Dev), and test (Test) instances for each dataset.

| Dataset | Task Category | Classes | Train | Dev | Test |
|---------|---------------|---------|-------|-----|------|
| IMDB    | Sentiment Classification | 2       | 20,000 | 5,000 | 25,000 |
| SST-2   | Sentiment Classification | 2       | 67,349 | 872  | 1,821 |
| RTE     | Natural Language Inference | 2       | 2,490  | 277  | 2,985 |
| QNLI    | Question Answering / Natural Language Inference | 2       | 104,743 | 5,463 | 5,463 |

TABLE 2. Hyper-parameter setting. Description of hyperparameter settings used for generating and retraining. Since the generating and retraining were performed separately for each dataset, we used different hyperparameters for each training.

| Process | Hyper-Parameter | IMDB | SST-2 | QNLI | RTE |
|---------|----------------|------|-------|------|-----|
| Generating | batch size | 8 | 16 | 8 | 16 |
| | dropout | 0.3 | 0.3 | 0.3 | 0.1 |
| | adversarial steps | 2 | 2 | 2 | 2 |
| | $\alpha$ | 1.25e-4 | 1.25e-4 | 1.25e-4 | 1.25e-4 |
| | $\epsilon$ | 1e-2 | 1e-2 | 1e-2 | 1e-2 |
| Retraining | batch size | 8 | 16 | 8 | 16 |
| | dropout | 0.3 | 0.3 | 0.35 | 0.1 |
| | $E_{\text{start}}$ | 12 | 5 | 2 | 65 |
| | $E_{\text{end}}$ | 16 | 13 | 6 | 70 |
| | retraining size | $\times 5$ | $\times 9$ | $\times 5$ | $\times 6$ |

TABLE 3. Classification accuracy of three methods for each dataset.

| Model | IMDB | SST-2 | QNLI | RTE |
|-------|------|-------|------|-----|
| BERT$\text{BASE}$ (Implemented) | 92.68 | 91.97 | 91.01 | 70.40 |
| PGD-Adversarial (Implemented) | 92.99 | 91.74 | 91.41 | 71.48 |
| Augmented (Ours) | 93.17 | 92.66 | 91.27 | **71.84** |

The dropout rate used during the generating process, was set to 0.1 or 0.3, and the number of adversarial steps was 2 or 3. The value of $\alpha$ that determines the size of the update when solving the inner maximization problem in the generating process was set to 1.25e-4, and the value of $\epsilon$ that determines the size of a norm boundary, was set to 1e-2. In the retraining process, dropout was set to 0.1, 0.3, or 0.35, and the size of the retraining dataset was set to five or nine times the size of the original data. The detailed hyperparameter settings are presented in Table 2. Fine-tuning, PGD-based adversarial training, generating, and retraining were conducted independently for each dataset. The embeddings used in all experiments were sub-word embeddings that are commonly used in BERT.

C. CLASSIFICATION PERFORMANCE ON THE ORIGINAL DATASETS

Table 3 presents the classification accuracies of the three methods for each dataset. The fine-tuning [2], PGD-based adversarial training [25], and the proposed method are referred to as BERT$\text{BASE}$, PGD-Adversarial, and Augmented, respectively. In the table, the test accuracy was reported for the IMDB dataset, whereas development accuracy [51] was reported for the SST-2, RTE, and QNLI datasets. This was because the SST-2, RTE, and QNLI datasets belonged to the GLUE benchmark [51], and thus the test accuracy could be evaluated using the online server after performing all other tasks belonging to the GLUE benchmark. For a fair comparison, we re-implemented the methods BERT$\text{BASE}$ and PGD-Adversarial and used them for all datasets. The same training, test (IMDB), and development (SST-2, QNLI, and RTE) datasets were used for all scenarios. Note that the test or development datasets were not involved in the training process including the generating and retraining processes. As shown in Table 3, the proposed method outperformed both the original fine-tuned model (BERT$\text{BASE}$) and PGD-Adversarial for three datasets (IMDB, SST-2, and RTE). In the QNLI dataset, PGD-Adversarial yielded the highest accuracy followed by the proposed Augmented method.

D. CLASSIFICATION PERFORMANCE ON SAMPLED DATASETS

As can be seen from Table 3, the proposed Augmented method showed lower performance than PGD-Adversarial on the QNLI dataset. Note that since the QNLI dataset has the largest training dataset among the datasets used in this paper, the risk of overfitting caused by insufficient training examples might be lower for the QNLI dataset than the other three datasets. Hence, we conducted an additional experiment to investigate the performance of the three methods with regard to the number of training examples for the QNLI dataset.
We randomly sampled the 10%, 20%, and 30% of training instances from the QNLI training set, and the augmentation was repeated five times. Except for the training dataset size, the same experimental settings including the hyperparameters were used for all three methods.

Figure 4 shows the performance of three methods according to the sampling ratio of the QNLI dataset. **BERT** yielded the accuracy of 85.26%, 88.21%, and 88.82% for the 10%, 20%, and 30% sampled dataset, respectively. **PGD-Adversarial** recorded 86.56%, 88.16%, and 89.29% for three sampling ratios, thus exhibited a 1.3%p performance improvement for 10% sampled dataset and 0.05%p performance degradation for 20% sample dataset compared to **BERT**. In contrast to the performance degradation of **PGD-Adversarial** for a particular sampling ratio, the proposed **Augmented** yielded the accuracy of 88.64%, 89.16%, and 89.9%, outperforming both **BERT** and **PGD-Adversarial** for all three sample ratios. In summary, the proposed method, **Augmented**, record a performance improvement of more than 0.95%p compared to **BERT** which fine-tuning is applied, when the size of the training dataset was small. In the case of **PGD-Adversarial**, a performance improvement was recorded compared to **BERT** at sampling ratios of 10% and 30%, but performance degradation was recorded compared to **BERT** at a sampling ratio of 20%. Therefore, it could be concluded that the proposed method, **Augmented**, worked more stably than **PGD-Adversarial**, which has a relatively large variation in performance improvement depending on the experimental conditions. These experimental results supported our claim that the proposed method, **Augmented**, could effectively and stably prevent overfitting when the size of the training dataset was small.

**E. COMPARISON WITH TOKEN LEVEL TEXT AUGMENTATION**

As specified in Section IV-C and Section IV-D, we conducted comparative experiments with **BERT** and **PGD-Adversarial**. However, since our proposed method, **Augmented**, is not very simple, we performed a comparative experiment with token level text augmentation, which can be performed more conveniently.

As a token level text augmentation method for comparison, **EDA** [20] was used to change certain words constituting text data into synonyms, insert a synonym of a random word in text data, change the position of two randomly selected words, or delete certain words with a random probability. In the case of **EDA**, sub-word embedding was formed after applying the method at the word level. For **EDA**, we set synonym replacement, which replaces a certain percentage of words with synonyms in the text data, to 0.05, and random deletion, which deletes a certain percentage of words from text data, to 0.1. The size of the augmented dataset was the same as the retraining size for each dataset in Table 2. The experimental results are listed in Table 4.

The proposed method, **Augmented**, outperformed **EDA** for IMDB, QNLI, and RTE dataset. Although **Augmented** recorded lower performance in SST-2 compared to **EDA**, **EDA** recorded a significant performance drop of more than 8%p in the RTE dataset compared to that of **Augmented**. In addition, **EDA** recorded improved or the same performance as that of **BERT** as shown in Table 3, in the IMDB and SST-2 datasets that employ single text as input, but recorded lower performance than **BERT** in the QNLI and RTE datasets that employ two texts as input. This suggested that the proposed method was more suitable for general use for various datasets than the **EDA** method.

**F. ANALYSIS ON PSEUDO-LABEL**

In this section, we demonstrate the validity of assigning the same label of the input text data to the adversarial embeddings as pseudo-labels. First, we visually analyze whether **BERT** embedding and the generated adversarial embedding are positioned close in the embedding space, such that adversarial embedding preserves semantic elements of the original input embedding. We used t-SNE [53] to reduce the dimension of the original and generated adversarial embeddings so that they can be visualized in a 2-dimensional space. For input embedding and adversarial embeddings, we used the QNLI dataset with the sampling ratio of 20% used for the experiment in Section IV-D. Sub-word tokenizing was applied to the input text, while question and context were connected using a special token [SEP]. In the visualization result, each point signifies the embedding for sub-word tokens including special tokens such as [CLS] and [SEP].
Figure 5 and Figure 6 each shows the result of visualizing (a) original input embedding and (b) the original input embedding along with the five adversarial embeddings generated for each sub-word token for different input texts. Figure 5 and Figure 6 both show that the distance between original input embedding and the corresponding adversarial embedding is very close in which it can be visually inspected that clusters are formed with respect to the sub-word token corresponding to original embedding and adversarial embedding. In most cases, the distance between the original input embedding and the corresponding adversarial embedding with respect to a specific token is closer than the distance
TABLE 5. The result of extrinsic analysis. The cosine similarities and average cosine similarity of original embedding and each adversarial embedding.

| Method | Dataset | 1st Augmentation | 2nd Augmentation | 3rd Augmentation | 4th Augmentation | 5th Augmentation | Average |
|--------|---------|------------------|------------------|------------------|------------------|------------------|---------|
| BERT_{BASE} | IMDB    | 0.9596           | 0.9599           | 0.9588           | 0.9597           | 0.9594           | 0.9595  |
|         | SST-2   | 0.9973           | 0.9970           | 0.9969           | 0.9972           | 0.9972           | 0.9971  |
|         | QNLI    | 0.9979           | 0.9979           | 0.9979           | 0.9979           | 0.9979           | 0.9979  |
|         | RTE     | 0.9873           | 0.9878           | 0.9863           | 0.9861           | 0.9878           | 0.9871  |
| Augmented | IMDB    | 0.9641           | 0.9640           | 0.9637           | 0.9639           | 0.9638           | 0.9639  |
|         | SST-2   | 0.9997           | 0.9998           | 0.9998           | 0.9998           | 0.9998           | 0.9998  |
|         | QNLI    | 0.9987           | 0.9987           | 0.9992           | 0.9989           | 0.9990           | 0.9989  |
|         | RTE     | 0.9967           | 0.9971           | 0.9953           | 0.9949           | 0.9973           | 0.9963  |

FIGURE 7. Classification performance with regard to the random label ratio.

to the closest sub-word token embedding with respect to the original input embedding. Therefore, a perturbation does not have a sufficient norm to convert the input embedding for a specific token into an embedding of another token. Consequently, an adversarial embedding positioned very close from the original input embedding in the embedding space can be assumed to maintain the characteristics of the original input embedding fairly well, and therefore, the label of the input text can be assigned to the sequence of adversarial embedding as a pseudo-label. Furthermore, since an LM is trained using the adversarial embedding with the same label of the original input embedding, the LM is trained to predict the sequence of adversarial embeddings to the same class of the original text. This means that the LM makes the same prediction for different embedding with subtle changes. Consequently, as shown in Figure 3, the region of the original input embedding can be expanded to the region where adversarial embedding is present, and a more generalized embedding space can be obtained for noise-injected embeddings.

In addition, we conduct another experiment with the RTE dataset to verify the appropriateness of assigning the pseudo-label to the sequence of adversarial embedding as the same label of the input text data. In this experiment, the pseudo-label of adversarial embedding generated during the generating process was converted to a random label at a certain ratio, and the LM was retrained. Four random label conversion ratio were used: 10%, 20%, 30%, and 40%.

Figure 7 shows the graph of the changes in the performance according to the random label ratio. For the RTE dataset, the proposed method using the pseudo-labels achieved an accuracy of 71.84%. However, it was dropped significantly to 64.62%, 63.54%, 64.26%, and 63.18%, when the random label ratio is increased from 10% to 40%, respectively. As we randomly assigned the labels in the experiment, there were some cases where the randomly assigned label matched the original pseudo-labels, which resulted in a non-monotonic decrease of accuracy with regard to the random label ratio. However, it is still confirmed that the randomly assigned labels for the adversarial embeddings significantly degenerate the performance of the retraining model regardless of the random label ratio. Based on this experiment, it can be supported that assigning the label of the original embedding to the sequence of adversarial embeddings is an appropriate strategy to improve the generalization ability of LM.

G. ANALYSIS ON ADVERSARIAL EMBEDDING

In section IV-F, we visually demonstrate that adversarial embeddings are generated at close distances to the original embedding in the embedding space. In this section, we perform an extrinsic analysis to demonstrate the prediction model output similarity between the original and adversarial embeddings. We first generated adversarial embeddings by performing the generating process using the evaluation dataset and not the training datasets. These adversarial embeddings were then used as the input for each LM that had been trained for the target task. The outputs of the LM with the adversarial embeddings were compared with the outputs of the same LM with the original embedding of the evaluation dataset.

The IMDB, SST-2, QNLI, and RTE datasets were used for the experiment, and the generating process was performed on the test dataset (IMDB) or the development datasets (SST-2, QNLI, and RTE) to generate five adversarial embeddings for each of the original data. To generate adversarial embeddings for the entire dataset, we did not apply mini-batch sampling during the generating process. The generating process was performed in ten epochs, and adversarial
embeddings generated from the fifth epoch to the ninth epoch were used. BERT\textsubscript{BASE} and Augmented were used as the LMs. We computed the cosine similarity between the original and adversarial embeddings, as shown in (3).

\[
\text{Cosine Similarity}_i = \frac{\text{Prob}_{\text{Original}, i} \cdot \text{Prob}_{\text{Augmented}, i}}{||\text{Prob}_{\text{Original}, i}|| \cdot ||\text{Prob}_{\text{Augmented}, i}||}
\]

where \text{Prob}_{\text{Original}, i} and \text{Prob}_{\text{Augmented}, i} denote the predicted probabilities of the positive class of the original embeddings in the evaluation set and those of the \(i\)th augmented embeddings in the evaluation set, respectively.

Table 5 presents the cosine similarities between the original embedding and each adversarial embedding in addition to their average in the last column. In all cases, the original and adversarial embeddings showed a higher cosine similarity greater than 0.95. Therefore, it could be confirmed that the original and adversarial embeddings behaved similarly in the downstream task. In addition, for all datasets, the average cosine similarity from Augmented was higher than the average cosine similarity from BERT\textsubscript{BASE}. This confirmed that the retrained LM recognized adversarial embedding more similarly to the original embedding compared to the fine-tuned LM. Therefore, it could be concluded that the retrained LM was more generalized to noisy embedding.

V. CONCLUSION

In this paper, we proposed a text embedding augmentation method for improving the generalization performance while preventing overfitting, which may occur when fine-tuning an LM having a large number of parameters. As proposed by Zhu \textit{et al.} [25], a perturbation was trained to increase the loss for input embedding of an LM within a certain norm boundary, and the LM parameters were trained to reduce the LM loss for adversarial embedding in which the perturbation is added to the input embedding. Then, we assigned the label of original input text data to the generated adversarial embeddings in adversarial training to augment the original dataset. Subsequently, the constructed pseudo-labeled adversarial embedding was employed as the labeled input embedding for fine-tuning a separate LM. In the proposed method, the adversarial embedding was generated by adding a perturbation trained within a very small norm boundary around the input embedding, thus it can both preserve the characteristics of the input embedding and increase the robustness to noises. Consequently, the generalization performance of LM can be improved by LM through repeated training of the adversarial embedding generated using different model parameters. Experimental results on four natural language understanding datasets show that the proposed text embedding augmentation method outperforms the benchmark models in most cases.

Based on the favorable experimental results, the proposed text augmentation method can be extended to various NLP downstream tasks. The datasets considered in this paper have document-level labels. In other words, only one label is assigned to the entire input document. However, there are other NLP tasks with more sophisticated label information for which token-level labels are required, such as named entity recognition and question-answering. Hence, the proposed method should be verified whether it can prevent model overfitting by adversarial embedding-based text augmentation through more diverse NLP tasks and datasets.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in Proc. Annu. Conf. Neural Inf. Process. Syst., Long Beach, CA, USA, 2017, pp. 5998–6008.
[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol., Minneapolis, MN, USA, vol. 1, Jun. 2019, pp. 4171–4186. [Online]. Available: https://www.aclweb.org/anthology/N19-1423
[3] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, “RoBERTa: A robustly optimized BERT pretraining approach,” 2019, arXiv:1907.11692.
[4] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, “XLNet: Generalized autoregressive pretraining for language understanding,” in Proc. Annu. Conf. Neural Inf. Process. Syst., Vancouver, BC, Canada, 2019, pp. 5756–5766.
[5] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H. Hon, “Unified language model pre-training for natural language understanding and generation,” in Proc. Annu. Conf. Neural Inf. Process. Syst., Vancouver, BC, Canada, 2019, pp. 13042–13054.
[6] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, “ALBERT: A lite BERT for self-supervised learning of language representations,” in Proc. 8th Int. Conf. Learn. Represent., Addis Ababa, Ethiopia, 2020, pp. 1–16.
[7] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, “BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,” in Proc. 58th Annu. Meeting Assoc. Comput. Linguistics, 2020, pp. 7871–7880.
[8] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” J. Mach. Learn. Res., vol. 21, no. 140, pp. 1–67, 2020.
[9] J. Hewitt and C. D. Manning, “A structural probe for finding syntax in word representations,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Human Lang. Technol., vol. 1, Minneapolis, MN, USA, 2019, pp. 4129–4138.
[10] E. Reif, A. Yuan, M. Wattenberg, F. B. Viégas, A. Coenen, A. Pearce, and B. Kim, “Visualizing and measuring the geometry of BERT,” in Proc. Annu. Conf. Neural Inf. Process. Syst., Vancouver, BC, Canada, 2019, pp. 8592–8600.
[11] H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and T. Zhao, “SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization,” in Proc. 58th Annu. Meeting Assoc. Comput. Linguistics, Jul. 2020, pp. 2177–2190.
[12] X. Ying, “An overview of overfitting and its solutions,” J. Phys., Conf. Ser., vol. 1168, Feb. 2019, Art. no. 020202.
[13] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org
[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” J. Mach. Learn. Res., vol. 15, no. 56, pp. 1929–1958, 2014.
[15] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” 2016, arXiv:1607.06450.
[16] J. Kukačka, V. Golkov, and D. Cremers, “Regularization for deep learning: A taxonomy,” 2017, arXiv:1710.10686.
[17] P. Liu, X. Wang, C. Xiang, and W. Meng, “A survey of text data augmentation,” in Proc. Int. Conf. Comput. Commun. Netw. Secur. (ICCNs), Aug. 2020, pp. 191–195.
[18] C. Shorten and T. M. Khoshgoftaar, “A survey on image data augmentation for deep learning,” J. Big Data, vol. 6, no. 1, pp. 1–48, Dec. 2019.
Y. Cheng, L. Jiang, and W. Macherey, “Robust neural machine translation,” in Proc. 54th Ann. Meeting Assoc. Comput. Linguistics, vol. 1, Berlin, Germany, 2016, pp. 1715–1725.

C. Xie, M. Tan, B. Gong, J. Wang, A. L. Yuille, and Q. V. Le, “Adversarial examples improve image recognition,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Seattle, WA, USA, Jun. 2020, pp. 816–825.

A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, “Learning word vectors for sentiment analysis,” in Proc. 49th Ann. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol., Portland, OR, USA, Jun. 2011, pp. 142–150.

R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, “Recursive deep models for semantic compositionality over a sentiment treebank,” in Proc. Conf. Empirical Methods Natural Lang. Process., Seattle, WA, USA, Oct. 2013, pp. 1631–1642.

J. Wei and K. Zou, “EDA: Easy data augmentation techniques for boosting performance on text classification tasks,” in Proc. Conf. Empirical Methods Natural Lang. Process., 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP), Hong Kong, 2019, pp. 6381–6387.

R. Sennrich, B. Haddow, and A. Birch, “Neural machine translation of rare words with subword units,” in Proc. 54th Ann. Meeting Assoc. Comput. Linguistics, vol. 1, Berlin, Germany, 2016, pp. 1715–1725.

C. Xie, M. Tan, B. Gong, J. Wang, A. L. Yuille, and Q. V. Le, “Adversarial examples improve image recognition,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Seattle, WA, USA, Jun. 2020, pp. 816–825.

A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, “Learning word vectors for sentiment analysis,” in Proc. 49th Ann. Meeting Assoc. Comput. Linguistics, Hum. Lang. Technol. Portland, OR, USA, Jun. 2011, pp. 142–150.

R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, “Recursive deep models for semantic compositionality over a sentiment treebank,” in Proc. Conf. Empirical Methods Natural Lang. Process., Seattle, WA, USA, Oct. 2013, pp. 1631–1642.

J. Dagan, O. Glickman, and B. Magnini, “The PASCAL recognising textual entailment challenge,” in Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Textual Entailment (Lecture Notes in Computer Science), vol. 3944. Berlin, Germany: Springer, 2006, pp. 177–190.

H. R. Bar, D. Ido, D. Bill, F. Lisa, G. Daniło, M. Bernardino, and S. Idan, “The second PASCAL recognising textual entailment challenge,” in Proc. 2nd PASCAL Challenges Workshop Recognising Textual Entailment, 2006, pp. 1–9.

D. Giampiccolo, B. Magnini, I. Dagan, and B. Dolan, “The third PASCAL recognising textual entailment challenge,” in Proc.ACL-PASCAL Workshop Textual Entailment & Paraphrasing, 2007, pp. 1–9.

B. Magnini, I. Dagan, H. T. Dang, and D. Giampiccolo, “The fifth PASCAL recognising textual entailment challenge,” in Proc. Second Text Anal. Conf. (TAC), Gaithersburg, MD, USA, 2009, pp. 1–18.

P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “SQuAD: 100,000+ questions for machine comprehension of text,” in Proc. Conf. Empirical Methods Natural Lang. Process., Austin, TX, USA, 2016, pp. 2383–2392.

A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, “GLUE: A multi-task benchmark and analysis platform for natural language understanding,” in Proc. EMNLP Workshop BlackboxNLP, Analyzing Interpreting Neural Nerv. NLP, New Orleans, LA, USA, 2018, pp. 1–20.

T. Wolf, J. Chaumond, L. Debate, V. Sanh, C. Delangue, A. Moi, and P. Cistac, “Transformers: State-of-the-art natural language processing,” in Proc. Conf. Empirical Methods Natural Lang. Processing, Syst. Demonstrations, 2020, pp. 38–45.

L. van der Maaten and G. Hinton, “Visualizing data using t-SNE,” J. Mach. Learn. Res., vol. 9, pp. 2579–2605, Nov. 2008.

PILSUNG KANG received the B.S. and Ph.D. degrees in industrial engineering from Seoul National University. He is currently pursuing the M.S. degree in industrial and management engineering at Korea University, Republic of Korea. His research interests include developing machine learning algorithms for both structured data and unstructured data (image, video, and text) and applying them to solve engineering and business problems, such as fault classification in manufacturing, abnormal behavior detection from system logs, and sentiment classification from news and review texts.

MYEONGSUP KIM received the B.S. degree in industrial engineering from Konkuk University. He is currently pursuing the M.S. degree in industrial and management engineering at Korea University. His research interests include developing machine learning algorithms for both structured data and unstructured data (image, video, and text) and applying them to solve engineering and business problems, such as fault classification in manufacturing, abnormal behavior detection from system logs, and sentiment classification from news and review texts.