IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 3, 2022, accepted May 30, 2022, date of publication June 8, 2022, date of current version June 13, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3180830

SciDeBERTa: Learning DeBERTa for Science Technology Documents and Fine-Tuning Information Extraction Tasks

YUNA JEONG, (Member, IEEE), AND EUNHUI KIM, (Member, IEEE)
Korea Institute of Science and Technology Information, Daejoen 34141, Republic of Korea

Corresponding author: Eunhui Kim (ehkim@kisti.re.kr)

This work was supported by the Korea Institute of Science and Technology Information under Grant K-22-L04-C07-S01.

ABSTRACT Deep learning-based language models (LMs) have transcended the gold standard (human baseline) of SQuAD 1.1 and GLUE benchmarks in April and July 2019, respectively. As of 2022, the top five LMs on the SuperGLUE benchmark leaderboard have exceeded the gold standard. Even people with good general knowledge will struggle to solve problems in specialized fields such as medicine and artificial intelligence. Just as humans learn specialized knowledge through bachelor's, master's, and doctoral courses, LMs also require a process to develop the ability to understand domain specific knowledge. Thus, this study proposes SciDeBERTa and SciDeBERTa(CS) as a pre-trained LM (PLM) specialized in the science technology domain. We further pretrain the DeBERTa, which was trained with a general corpus, with the science technology domain corpus. Experiments verified that SciDeBERTa(CS) continually pre-trained in the computer science domain achieved 3.53% and 2.17% higher accuracies than SciBERT and S2ORC-SciBERT, respectively, which are science technology domain specialized PLMs, in the task of recognizing entity names in SciERC dataset. In the JRE task of the SciERC dataset, SciDeBERTa(CS) demonstrated a 6.7% higher performance than baseline SCIIE. In the Genia dataset, SciDeBERTa achieved the best performance compared to S2ORC-SciBERT, SciBERT, BERT, DeBERTa and SciDeBERTa(CS). Furthermore, re-initialization technology and optimizers after Adam were explored during fine-tuning to verify the language understanding of PLMs.

INDEX TERMS Deep neural network, domain specific language model, fine-tuning, natural language processing.

I. INTRODUCTION

One of the recent views in language models (LMs) is that increasing the model size and the training dataset size has a positive effect on the accuracy performance of downstream tasks [1]. Consequently, the size of LM and amount of training data are increasing exponentially. The model size of Megatron-Turing NLG [2], GoPher [3] of DeepMind and ERNIE 3.0 [4] are as large as 530B, 280 B and 175 B, respectively. Recently, a model size of Google's PaLM [5] was 540 B.

Based on the standard benchmark SuperGLUE [6] for LMs, the five models of SS-MOE [7], Turing NLR v5 [8], ERNIE 3.0 [4], T5+UDG [9]], and DeBERTa [10] have exceeded the human baseline as of 2022. These five models have sizes of 269 B, 5.4 B, 10 B, 11 B, and 1.5 B, respectively. As shown in the benchmark results, such large LMs achieve successful results in most NLP tasks in the general domain. However, performance is limited in specialized domains beyond the general one.

This study aims to develop a large-capacity LM that has improved natural language understanding (NLU) performance in the science technology domain. The most intuitive method for domain-specific LMs is pre-training with domain-specific data. We initialize the pre-trained LM (PLM) with parameters trained on the general domain, and then train continually on domain-specific data. SciBERT [11] and S2ORC-SciBERT [12] are representative PLMs specialized in science and technology based on BERT [13]. We base our work on the DeBERTa to consider not only the global context of the input but also the local context. Also, DeBERTa has the smallest model size among the top five models of the SuperGLUE leaderboard; at 78 GB, its data size is also relatively small. However, it has excellent performance in relation to its model

The associate editor coordinating the review of this manuscript and approving it for publication was Bing Li.

size and training data size. We further analyze fine-tuning optimization techniques of the NLU tasks in the science technology domain.

The contributions of this study are as follows:

- We present SciDeBERTa and SciDeBERTa(CS), a PLM specialized to the science and technology domain. It has improved performance by continually training the DeBERTa [10] with the science technology corpus; S2ORC used for training is a dataset that covers broad scientific and technological fields, so it is relatively less biased to a specific field. SciDeBERTa (CS), which was trained by continual learning only with Computer Science(CS) domain abstract, confirmed good performance in the information extraction task of SciERC composed of the AI Society dataset. SciDeBERTa, trained by continual learning with S2ORC whole domain abstract data, confirmed good performance in information extraction task of Genia dataset.
- The following machine learning techniques were applied to the information extraction tasks in science technology domain based on PLM to improve performance.
  - The re-initialization suppresses the parameter overfitting problem that occurs with task data that are relatively smaller than the pre-training data. For the base model size consisting of 12 layers, the best performance in DeBERTa is achieved when reinitializing the 10th layer. This is a different result from BERT showing the best performance in reinitialization in the 12th layer. We analyze and present theoretically and experimentally the cause of performance improvement through parameter re-initialization.
  - We analyze the most suitable optimizer for information extraction tasks in the science and technology domain by theoretically and experimentally comparing AdamW [14], AdamP [15], and RAdam [16], which are the latest optimizers proposed after Adam [17]. In the information extraction task, we confirm that the effectiveness of RAdam without the learning scheduler is superior to that of AdamW and AdamP with slanted triangular scheduler.

This paper is organized as follows. In Section II, we briefly review the latest trends in PLMs, domain-specific PLMs, and fine-tuning techniques for NLU tasks. Section III describes the features of DeBERTa, which is the basis of SciDeBERTa, and the training data for the scientific domain-specific PLM. Section IV analyzes the re-initialization and optimization techniques that were used in the fine-tuning stage to improve the NLU performance. In Section V, we report the experimental results of SciDeBERTa and fine-tuning techniques introduced in Sections III and IV, followed by a conclusion section.

II. RELATED WORK

In this section, we review the previous works on domain-specific PLMs and efficient fine-tuning techniques for natural language understanding.

A. RECENT TREND OF TRANSFORMER MODEL

We summarize the previous works on transformer-based [18] advanced LMs and pre-training corpus for PLMs.

There are two mainstreams of transformer-based LMs: BERT [13], which consists of encoder blocks, and GPT [19], which consists of decoder blocks. BERT improves performance specialized in NLU such as sentence and word classifications, whereas GPT improves performance specialized in natural language generation. BERT uses two methods in pretraining: masked language modeling (MLM), which predicts the randomly masked tokens in input sentences and next sentence prediction (NSP), which matches the order of sentences.

BERT-based PLMs have been enhanced into various models with improved pre-training tasks. SpanBERT [20] improved MLM to predict spans instead of tokens that are relatively easy to solve. StructBERT [21] and ERNIE 2.0 [22] have changed the pre-training task to predict the sentence order of several sentences instead of two sentences. RoBERTa [19] separates each document and uses doc-sentence to sample inputs from only one document to improve context representation understanding by training consecutive sentences in the same document.

B. LANGUAGE MODELING ON SCIENTIFIC TECHNOLOGY DOCUMENTS

General LMs based on transformers solve downstream tasks using two kinds of datasets in each training process. First, a large amount of unlabeled text data is pretrained by self-supervised learning, and the model acquires a universal language representation. The trained knowledge is transferred by fine-tuning a PLM to the target data. The target data is labeled task data, and its size is relatively small compared to the pre-training data.

In the pretraining step, a general domain corpus is used for extracting knowledge that can be generally useful in NLP tasks. BERT has been pretrained with 13 GB of plain texts in total consisting of 800 M and 2,500 M words collected from the BooksCorpus [23] and English Wikipedia, respectively. XLNet [24] and RoBERTa [19] optimized BERT based on the observation that BERT was underfitted. They trained a model much longer with a larger batch size on more data.

The PLM shows an improved task performance when the gap is small between the corpuses used in pretraining and fine-tuning. The BooksCorpus and English Wikipedia data used in pretraining the BERT have few noises (for example, few spelling mistakes) and use formal writing style. Therefore, the PLM trained on these data shows good performance in most NLP task benchmarks and leaderboards that have similar characteristics. However, these models find it difficult to achieve good performance in social media conversations, product reviews, and community posts, which have many noises and are informal. This is particularly true if the target domain include technical terms that do not belong to the general language domain, such as financial, legal, biomedical, and scientific texts. Thus, TweetBERT, FinBERT, LegalBERT, BioBERT, PubMedBERT, and SciBERT have been researched as specialized LMs that pretrain the BERT with

a specific domain corpus instead of a general domain corpus [11], [25]–[29].

The pretraining data of SciBERT, the representative PLM in the science and technology field is composed of 82% biomedical domain and 18% computer science domain with 3.2B tokens. The S2ORC [12] dataset, which was released later, collected data in a more balanced manner in more diverse fields of science and technology. Among the models specializing in the science domains, S2ORC-SciBERT, which has been pretrained with 16.4 B tokens, shows better performance in processing tasks in the science and technology field than SciBERT.

BioBERT [28] and PubMedBERT [29], which are state-of-the-art models for biomedical NLP tasks, were both pretrained with biomedical domain text data collected from PubMed and PubMed Central (PMC). PubMed and PMC are databases developed and maintained by the National Library of Medicine. PubMed provides citations of biomedical journals together with abstracts, and PMC archives full-text articles. BioBERT uses a continual pretraining method that additionally pretrains the general BERT with biomedical articles from the general standpoint that the knowledge provided by a general domain LM would still be useful in biomedicine. In practice, continual pretraining works effectively when there is a small amount of domain-specific data for pretraining. PubMedBERT considers that PuMed and PMC provide biomedical unlabeled text data sufficient to pretrain a general LM: 33 million abstracts in PubMed and 7.6 million articles in PMC. Hence, PubMedBERT randomly initializes all parameters of BERT and performs pretraining completely with biomedical in-domain texts only. PubMedBERT pretrained from scratch showed a better performance in some biomedical NLP tasks than BioBERT.

C. FINE-TUNING OF LANGUAGE MODELING

Various studies have been conducted to improve effectiveness in the process of fine-tuning a PLM to downstream tasks of a specialized domain. The most intuitive and widely used solution is to use the optimization technique. BERT showed improved performance compared to the conventional LM by using BERTAdam, a modified version of the ADAM [13]. BERTAdam plays the role of a warm-up for the learning rate by rescaling the learning rate for each epoch in place of the bias correction of Adam [17]. It improved stable training output without a separate warm-up for fine-tuning in the early stages of training [30]. In the following research, Xiong *et al.* used the Pre-Layer Normalization (pre-LN) neural network that places the normalization layer of the transformer block in front of the multi-head attention layer. The pre-LN structure showed stable training output in the early steps without a separate warm-up [31], [32]. On the other hand, bias correction plays the same role as warm-up in fine-tuning, and one study showed stable results by doing warm-up at the beginning of training and training for a long time [33]. The Adaptive Model Initialization (Admin) suggests that the cause of the initial instability of the PLM was the high dependency on residual connection [34]. Thus, additional parameters that adapt to the variance of outputs were applied to reduce residual dependency; it was concluded that determining the optimal hyperparameter is more important than a warm-up.

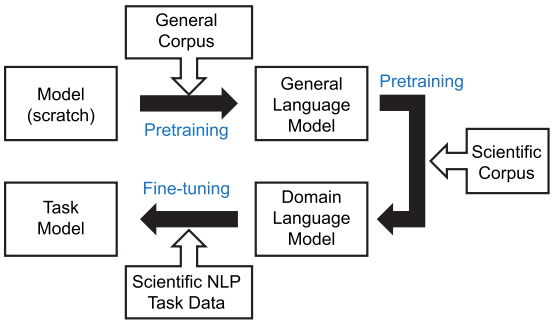

FIGURE 1. SciDeBERTa pretraining procedure for science technology domain.

One of the main problems in fine-tuning is the suppressing of overfitting resulting from domain gap and difference in size between the dataset for the pretraining of PLM and fine-tuning. T. Zhang *et al.* introduced layer reinitialization of neural networks as a method to solve overfitting of fine-tuning data learning for parameters over-fitted to pre-training data through PLM in specific BERT [30].

This study focuses on the optimizers AdamW [14], RAdam [16], and AdamP [15], which have improved the most widely used Adam optimizer based on L2 regularization, weight decay, and learning rate scheduler. The characteristics of these three optimizers are compared and the optimal algorithm for fine-tuning domain-specialized PLM is verified. Additionally, in the proposed study, the reinitialization method to suppress overfitting considering the characteristics of the SciDeBERTa model is explored through theory and experiment.

III. SciDeBERTa: DeBERTa AND SCIENCE TECHNOLOGY DATASET

We introduce SciDeBERTa, a PLM based on DeBERTa for the science domain. Figure 1 shows the training process of SciDeBERTa. We base our work on DeBERTa$_{base}$. SciDeBERTa is first initialized by DeBERTa, which has been pretrained with a general domain corpus. Next, additional pretraining is performed with a dataset of the science domain. Finally, the PLM is fine-tuned with the domain task data to resolve downstream tasks.

A. DeBERTa CHARACTERISTICS

DeBERTa [10] shows excellent performance in relation to its model size and training data size although it has the smallest model size among the top models that exceeded the human baseline in SuperGLUE. Therefore, the PLM specialized in the science technology domain proposed in this study is based on the DeBERTa base model. Similar to BERT and RoBERTa, DeBERTa [10] is an LM pretrained with a 78 GB of unlabeled English corpus collected from English Wikipedia, BookCorpus, OpenWebText, and Stories. DeBERTa has improved the performance of RoBERTa by applying the two techniques of disentangled attention mechanism and enhanced mask decoder.

Similar to most transformer-based LMs, the input representation of RoBERTa [19] is the sum of the token and absolute position embedding. DeBERTa uses a vector obtained by concatenating relative position embedding to token embedding as input. The attention weight of DeBERTa is obtained by disentangled attention mechanism; the disentangled attention is calculated using the cross attention between the queries and keys of the content (token) embedding of tokens and the relative position embedding. This calculation of the cross-attention score reflects the relative distance between tokens and considers the dependency among tokens, which is a local context that is difficult to reflect in the standard self-attention mechanism.

Similar to BERT and RoBERTa, DeBERTa also uses absolute position to consider the global context. The conventional LM sums the absolute position embedding to the input representation, but DeBERTa incorporates absolute position embedding by adding it to the input of the last n^{th} transformer layers before the softmax layer, denoted as enhanced mask decoder.

B. DATASET FOR PRE-TRAINING SciDeBERTa MODEL

SciBERT was pretrained using a corpus consisting of 82% biomedical domain and 18% computer science domain. In contrast, S2ORC-SciBERT was pretrained with BERT using a relatively balanced distribution of data in more diverse fields of study. Even though it was trained for a more general domain, S2ORC-SciBERT showed performance similar to or better than SciBERT in biomedical and computer science domain tasks. Based on these observations, SciDeBERTa uses data from all fields of study provided by S2ORC for additional pretraining so that it can be used for text mining in the general science domain.

S2ORC provides both abstracts and full texts of scientific papers. Unlike abstracts, full texts are not standardized and contain considerable noise. Hence, if full texts are to be used for training, the model needs to be trained with a larger amount of data for a longer time to sufficiently understand the knowledge contained in the texts. PubMedBERT [29] showed an experiment result where the model pretrained with 12 GB of abstracts had little difference in performance with a model trained with 128 GB of abstracts and full texts if the training time was the same. For longer training, PubMedBERT trained with full-text showed performance improvement on several downstream tasks. Nevertheless, SciDeBERTa uses only the abstracts of S2ORC in continual learning because performance degradation may occur without sufficient data and learning time, and field coverage should also be considered.

IV. FINE-TUNING WITH MULTIPLE INFORMATION EXTRACTION TASKS

GLUE and SuperGLUE are representative benchmarks of language understanding tasks. However, these benchmarks consist of the general language corpus, thus they are unsuitable for use in the evaluation of LMs trained with science and technology documents. One method for evaluating a PLM trained with science and technology data is to extract the knowledge information conveyed by the sentences. In other words, named entity recognition (NER), relation extraction (RE), and co-reference (Coref) resolution are performed as tasks to extract knowledge information.

The NER can be classified as a token (span) classification problem. There are two methods for relation extraction: sentence classification that extracts relations from all sentences (we call this method RE), and joint entity recognition and relation extraction that extracts entities and their relations (we call this method JRE). Coref resolution identifies the co-reference information among major entities or about entities through synonyms. In this study, we evaluate the fine-tuning performance through the NER, RE, JRE, and Coref resolution tasks of the SciERC dataset [35] and NER task of the Genia dataset [36].

Three tasks such as NER, JRE and Coref can be performed individually. However, the performance of the fine-tuning is improved by simultaneously performing three tasks. DyGIE++ [37] generates a span after receiving the embedding of a PLM as input, performs a weighted sum of losses by performing three types of tasks, and updates the span information according to the loss. It has been experimentally proven that the performance of NER and JRE tasks is improved when the span information is updated by conveying the Coref resolution information. This study verified the improved performance by exploring re-initialization and optimizer technology in NER, JRE and Coref tasks.

A. UPPER-LAYER RE-INITIALIZATION TO SOLVE OVER-FITTING

In Section II-C, we mentioned the re-initialization of training parameters by the resolution to suppress the overfitting. In order to decide which layer to apply re-initialization to, we first review previous studies on the representation characteristics by a layer of LMs.

Aken *et al.* revealed that it is the layer immediately before the last that determines the performance applying the probing method for each layer of BERT [38]. Clark *et al.* compared and analyzed the magnitude of the attention value of the layers for a specific token [39]. The results proved that in the low layer, the ''CLS'' token, which reveals the overall features of sentences shows a large attention value; in the middle layer, tokens such as ''SEP'' that classify two sentences show a large attention value; and in the top layer, frequent words such as ''.'' and '','' show a large attention value. The higher the layer, the more the features of the detailed and frequent tokens of the sentence are revealed. These research findings indirectly reveal that data changes can be an overfitting factor. The re-initialization of the top layer may resolve overfitting, and this study verifies this through experiments.

Re-initialization is also beneficial for eliminating the accumulated noise of residual connections. One block of a transformer-based model is generally composed of a residual connection and a normalization layer after a multi-head attention layer [19]. RadFord *et al.* successfully achieved performance improvement through a structural change from GPT [19] to GPT2 [40]. GPT2 [40] was composed of a pre-LN (normalization layer before each block) structure while increasing the number of layers compared to GPT and

Algorithm 1 Adam [17]: *Adaptive Moment Estimation*

Input: $\alpha(lr)$; β_1, β_2; $\theta_0(params)$; $f(\theta)$; ϵ
Init: $m_0 \leftarrow 0$ *1st moment;* $v_0 \leftarrow 0$ *2nd moment*
Output: *optimized parameter* θ_t

1: ***for*** $t = 1, \ldots$ ***do***
2: $\quad g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
3: $\quad m_t \leftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t$
4: $\quad v_t \leftarrow \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2$
5: $\quad \hat{m}_t \leftarrow m_t / (1-\beta_1^t)$
6: $\quad \hat{v}_t \leftarrow v_t / (1-\beta_2^t)$
7: $\quad \theta_t \leftarrow \theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$
8: ***end for***

composing each block as a residual connection. Moreover, additional layer normalization was applied even after the last self-attention, and re-initialization was applied to the depth at which residuals accumulate [40]. Composing a block of the pre-LN structure and re-initializing high layers have the dual effects of resolving the cumulative errors of the model and resolving the difference in detailed tokens owing to the data difference between the pre-training and fine-tuning dataset.

Combining the results of previous studies, the overfitting layer is concluded as the upper layers in which the neural network is repeatedly stacked [30]. However, in the case of DeBERTa, the relative positional embedding is equally stacked from the bottom to the top 3 layers, and in the case of the last two layers, the absolute positional embedding is applied in the same way as the existing BERT. As a result, BERT and DeBERT have different neural network characteristics. In the DeBERTa model, the overfitting layer is contextually the top three layers, and the last two layers can be interpreted as new layers. Experimental verification results also support this. Contrary to the fact that the last layer in BERT has a large overfitting effect upon reinitialization, DeBERTa confirms that the reinitialization layer helps in overfitting when the last 3rd layer is included.

B. EXPLORER OPTIMIZER

The Adam optimizer was developed in 2015 by combining the advantages of the adaptive gradient optimizer (Adagrad) and the root mean square optimizer (RMSProp) [17]. Adam increases the step size of the less-visited places and reduces the step size of the frequently visited places for balanced movement as Adagrad updates the parameter, and applies the exponential weighted moving average (EMA) of RMSProp to give weight to the recent gradient value. The Adam algorithm is shown in Algorithm 1. Lines 3 to 6 represent the first moment estimate, the second moment estimate, the first moment bias correction, and the second moment bias correction. When the equation in line 7 is differentiated, (1) is obtained. This is a proportional update $\frac{averagegradient}{\sqrt{(averagegradient)^2}}$, which is a characteristic of Adagrad, and the normalization value is bound by (2). The advantage of the Adam algorithm is the presence of a scale-invariant bounded norm; it is argued

Algorithm 2 RAdam [16]: *Rectified Adam*

Input: $\alpha_{t=1}^{T}(lr)$; β_1, β_2; $\theta_0(params)$; $f(\theta)$
Init: $m_0 \leftarrow 0$ *1st moment;* $v_0 \leftarrow 0$ *2nd moment,* $\rho_\infty \leftarrow 1/(1-\beta_2) - 1$ ***max length of the SMA***
Output: *optimized parameter* θ_t

1: ***for*** $t = 1, \ldots, T$ ***do***
2: $\quad g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$
3: $\quad m_t \leftarrow \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t$
4: $\quad v_t \leftarrow \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2$
5: $\quad \hat{m}_t \leftarrow m_t / (1-\beta_1^t)$
6: $\quad \rho_t \leftarrow \rho_\infty - 2t\beta_2^t/(1-\beta_2^t)$
7: $\quad$ ***if*** *the variance is tractable, i.e.,* $\rho_t > 4$ ***then***
8: $\quad\quad \hat{v}_t \leftarrow v_t / (1-\beta_2^t)$
9: $\quad\quad r_t \leftarrow \sqrt{\frac{(\rho_t-4)(\rho_t-2)\rho_\infty}{(\rho_t-4)(\rho_t-2)\rho_t}}$
10: $\quad\quad \theta_t \leftarrow \theta_{t-1} - \alpha r_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t}}$
11: $\quad$ ***else***
12: $\quad\quad \theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t$
13: $\quad$ ***end if***
14: ***end for***

that even if the gradient increases, the step size is bound, and stable optimization descent is possible.

$$\nabla_t = -\alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}} \tag{1}$$

$$\left\| \nabla_t \right\|_\infty \leq \begin{cases} \alpha \cdot \dfrac{1-\beta_1}{\sqrt{1-\beta_2}} & if\,(1-\beta_1) > \sqrt{1-\beta_2} \\ \alpha & otherwise \end{cases} \tag{2}$$

However, compared to SGD, Adam is weakly generalizable because, in SGD, L_2 regularization coincides with weight decay; however, in Adam, the weight decay rate becomes smaller than L_2 regularization. Additionally, optimizers usually add acceleration in the learning direction along with momentum, but if the size of momentum increases while less learning is done, learning will not reach the convergence point stably.

To improve the Adam optimizer and to achieve a precise regularization, AdamW separately applies weight decay and L_2 regularization [14]. Furthermore, when it is combined with the learning rate scheduler, improved performance can be obtained. For stable convergence, AdamP adjusts the momentum according to the normalization state [15]. However, the learning rate is delayed by approximately 8% owing to the additional calculation of the normalization.

The RAdam algorithm is shown in Algorithm 2 [16]. RAdam replaces the EMA of Adam with the simple moving average (SMA), which is widely used in economics, and adjusts the various sizes of the rectification term, which acts as the adaptive learning rate according to the Degree of Freedom (DoF) ρ. Line 6 of Algorithm 2 is the DoF ρ, which is the length of the SMA. If the variance is tractable according to the condition of line 7, the rectification term in line 9 adjusts the variance of the learning rate according to the

TABLE 1. Entities and relations in SciERC and Genia dataset.

dataset	knowlg	# elements	# tag
SciERC	Entities	Task, Method, Evaluation Metric, Material, Other Scientific Terms, Generic	8,094
	Relations	Used for, Part of, Hyponym of, Feature of, Evaluate for, Conjunction, Compare	6,319
Genia	Entities	DNA, RNA, protein, cell line, cell type	96,293

TABLE 2. Hyperparameters for pretraining SciDeBERTa with S2ORC.

Hyperparameter	Assignment	
	SciDeBERTa (CS)	SciDeBERTa
max training steps	12.5K	25.0K
batch size	2048	2048
learning rate	0.0005	0.0005
optimizer	Adam	Adam
weight decay	0.01	0.01
learning rate decay	linear	linear

TABLE 3. Hyperparameters for fine-tuning.

Hyperparameter	Assignment
pre-trained model	DeBERTa [10]
max span width	8
number of training epochs	50
optimizer lr	0.001
optimizer weight decay	0.0
embedder lr	5e-5
embedder weight decay	0.01
learning rate decay	slanted triangular (default)

DoF ρ. Unlike variance, which is tractable, that is, if $\rho_t \geq 4$, it is updated as shown in line 12. RAdam has the advantage of enabling stable training by replacing warm-up, which plays the role of variance reduction, with the variance adjustment of the rectification term.

In Section V, the experiments showed that all three models AdamW [14], AdamP [15], and RAdam [16] showed improved training result that was stable compared to Adam [17]. The three models AdamW, AdamP, and RAdam showed similar performance. The improved performance was confirmed because RAdam showed the advantage of stable training without being affected by the learning rate and learning rate scheduler.

C. DATASET FOR FINE-TUNING

The tasks for the experiments are the NER, JRE and Coref of SciERC and NER and Coref of GENIA.

SciERC is composed of annotated entity names, relations, and cross-references for the abstracts of 500 papers from 12 AI conferences according to the ACL RD-Tec 2.0 [41]. The entity names and related information of the SciERC dataset are not information about people and places that are used for general purposes, but elements to verify the knowledge information and system of scientific documents and comprise elements as shown in Table 1. SciERC is composed of 2,867 sentences; the total number of named entities is 8,094, and the total number of relations is 6,319. In other words, it is a relatively small dataset composed of less than 10,000 tagged entity names [35].

GENIA contains 2,000 abstracts from Medline articles [36]. It consists of a total of 18,545 sentences and 436,967 words. There are 96,293 entities, which are defined in terms of hierarchical structure, and 35 fine-trained entity categories. For ontology of Genia, please refer to Kim's thesis [36]. The subcategories are collapsed into five single labels (DNA, RNA, protein, cell line and cell type) as described in Table 1.

V. EXPERIMENTAL SETUP AND RESULTS

A. EXPERIMENTAL ENVIRONMENT AND IMPLEMENTATION DETAILS

We used one node that connects eight units of A100 GPU by the NVlink to pre-train the LM; the GPU RAM is 80 GB each, for a total of 640 GB. For fine-tuning, we configured the hardware of an Intel Core i7 machine with NVME SSD, 64 GB RAM, and two RTX 2080 GPUs; the GPU RAM is 24 GB each, for a total of 48 GB. Table 2 shows the hyperparameters for pretraining of SciDeBERTa with S2ORC. The hyperparameters for fine-tuning are shown in Table 3. We perform multi-task using DyGIE++ in fine-tuning. In SciERC, the loss weight ratios of NER, JRE, and Coref for each target task are (1.0, 0.0, 0.0) for NER, (0.2, 1.0, 1.0) for JRE, (0.5, 0.5, 1.0) for Coref, respectively. In Genia, only the target task is different, and the ratio of NER and Coref is the same as (1.0, 1.0).

Re-initialization, a technique for generalization, can be expected to improve performance in small datasets, which is the main cause of overfitting. This is the case with SciERC, which is a small task with less than 10,000 entities. On the other hand, there is almost no performance improvement for data of a certain size like Genia. Therefore, we applied re-initialization only to experiments on the SciERC task. Experiments on reinitialization and optimization use the DeBERTa model. The difference between SciDeBERTa and DeBERTa is the difference in training data, and the neural network structure is the same.

B. PERFORMANCE COMPARISON ACCORDING TO THE PLM AND DATASET

Table 4 and Table 5 show the pretraining step information and performance comparison results of PLMs specialized in the science and biomedical domain. The evaluation metric is average F1-score as described in equation (5) (i.e., in terms of both precision and recall, as in equations (3) and (4)). In equation (5), N is 5 in this study.

$$precision = \frac{true_{positive}}{true_{positive} + false_{positive}} \tag{3}$$

$$recall = \frac{true_{positive}}{true_{positive} + false_{negative}} \tag{4}$$

$$AverageF1 = \frac{2}{N}\sum_{n=1}^{N}\frac{precision \times recall}{precision + recall} \tag{5}$$

In the NER task of SciERC, comparing the results of base general domain LMs, DeBERTa, which improved BERT,

TABLE 4. Summary of pretraining step.

Model	Pretraining Step			
	Method	Corpus Type	Text Size	Dataset Type
BERT [13]	from scratch	general	3.3B tokens/16GB	–
DeBERTa [10]	from scratch	general	78GB	–
SciBERT [11]	from scratch	science	3.2B tokens	full-text
S2ORC-SciBERT [12]	from scratch	science	16.4B tokens	full-text
SciDeBERTa (CS)	continual learning	general+science	5.8GB	abstract
SciDeBERTa	continual learning	general+science	60.0GB	abstract

TABLE 5. Test performances (F1-score) of SciDeBERTa and PLMs specialized in science and technology fields.

Model	SciERC			GENIA	
	NER	JRE	Coref	NER	Coref
BERT [13]	67.9 ± 0.5	44.7 ± 0.8	55.0 ± 0.2	76.6 ± 0.2	44.9 ± 0.3
DeBERTa [10]	68.9 ± 0.6	44.6 ± 0.7	**59.3 ± 0.7**	77.8 ± 0.1	46.5 ± 0.8
SCIIE [35]	64.2*	39.3*	48.2*	–	–
SciBERT [11]	67.57*	–	–	78.0 ± 0.2	44.9 ± 1.0
S2ORC-SciBERT [12]	68.93* ± 0.19	–	–	78.1 ± 0.2	45.7 ± 0.2
SciDeBERTa (CS)	**71.1 ± 0.6**	**46.0 ± 0.8**	57.4 ± 0.6	77.8 ± 0.3	46.3 ± 0.4
SciDeBERTa	70.8 ± 0.5	45.5 ± 0.8	56.5 ± 0.8	**78.7 ± 0.3**	**47.0 ± 0.7**

† Our experimental results are average values of 5 runs.
† * denotes reported test results.

outperformed BERT even in the scientific domain task. Among the LMs specialized in science and technology, SciDeBERTa(CS) showed the best performance that was 3.53% higher than SciBERT, 2.17% higher than S2ORC-SciBERT and 0.3% higher than SciDeBERTa. Since SciDeBERTa(CS) was trained on abstract data from academic papers of computer science, it showed better performance than SciDeBERTa trained with other science and technology fields data in the SciERC dataset. In the SciERC JRE task, SciDeBERTa(CS) showed 6.7% higher performance than baseline SCIIE. However, In the SciERC Coref task, DeBERTa showed the best performance, with a 1.9% higher performance compared to SciDeBERTa(CS). In the SciERC dataset, all three tasks of NER, JRE and Coref, SciDeBERTa showed slightly lower performance than SciDeBERTa(CS). However, NER and Coref tasks in GENIA dataset, which is from medline articles, SciDeBERTa showed the best performance compared to other models. In the NER task of the GENIA dataset, SciDeBERTa showed 0.9% higher performance than SciDeBERTa(CS) and 2.1% higher performance than BERT. In the Coref task of the GENIA Dataset, SciDeBERTa demonstrated 0.7% higher performance than SciDeBERTa(CS) and 2.1% higher performance than BERT. As can be seen from Table 5, the above-mentioned terms are repeated in the coref task as pronouns. For this task, DeBERTa showed a difference of 1.6% to 4.3% compared to BERT, showing good performance. And in the coref task, it can be seen that DeBERTa trained as a general language model generally performs better than SciBERT, S2ORC-SciBERT and SciDeBERTa trained as a science and technology model.

C. PERFORMANCE COMPARISON BY RE-INITIALIZATION OF LAYERS

The models used in the experiments in Figure 2 and Figure 3 are all 12-layer base models, and the experiment was performed five times and the average value was taken in consideration of variance due to random initialization. The four kinds of tasks NER, RE, JRE, and Coref are used for fine-tuning tasks on the SciERC dataset. Figure 2 confirms that the last 12th layer re-initialization is most suitable for the overfitting prevention effect through the layer reinitialization method in SciBERT. Such result in re-initialization performance according to layer is consistent with the explanation in Section IV-A that more frequent words are expressed in the higher layer than in the main flow of context. However,

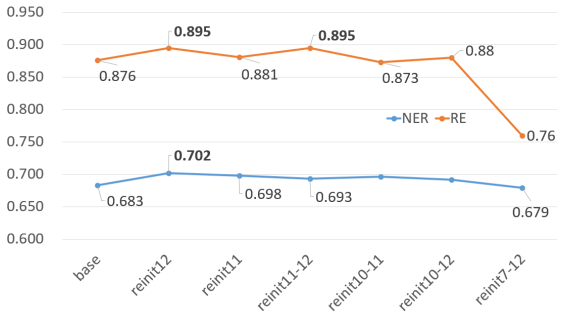

FIGURE 2. NER, RE task performance change by re-initialization applied layer; PLM is SciBERT [11].

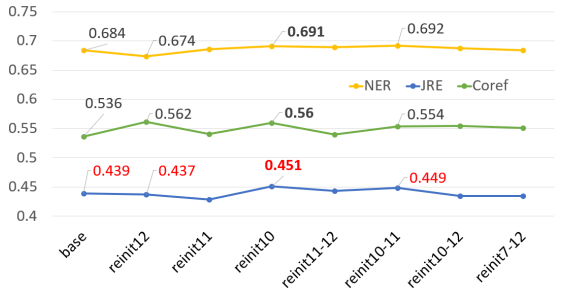

FIGURE 3. NER, JRE, Coref task performance change by re-initialization applied layer; PLM is DeBERTa [10] and target task is JRE. The loss-weight ratio of (NER, JRE, Coref) is (0.2, 1.0, 1.0).

as Figure 3 shows, the layer that reveals the overfitting prevention effect through re-initialization in the DeBERTa model is the 10th layer in the base PLM of 12 layers. As discussed earlier in Section IV-A, DeBERTa uses the same relative positional embedding from the lowest to the 10th layer, and uses absolute positional embedding for the 11th and 12th layers. In the case of BERT, all 12 layers use absolute positional embedding.

Figure 2 shows changes in the average F1 score for NER and RE tasks. As depicted in Figure 2, the average F1 performance improved approximately by 0.99% through the re-initialization of the last layer. Using the same method, the improvement of the average F1 performance by approximately 1.85% can be confirmed through the re-initialization of the last two layers for the RE task.

Figure 3 shows changes in the average F1 score for the NER, JRE, and Coref tasks by using DyGIE++. When using the DyGIE++ model, the target task is set as a JRE task. The ratios of NER, JRE, and Coref tasks when calculating loss are 0.2, 1.0, and 1.0, respectively. In Figure 3, the F1 score comparison of the reinitialization layer is based on the JRE task. Although the initialization of other layers is slightly more effective for NER and Coref, the JRE task is the target task, so it was taken as the standard. In the case where reinitialization was applied to the 10th layer (reinit10), compared to the case where reinitialization was not applied (base), NER, JRE, and Coref improved by 0.7%, 1.2%, and 2.4% F1 scores, respectively.

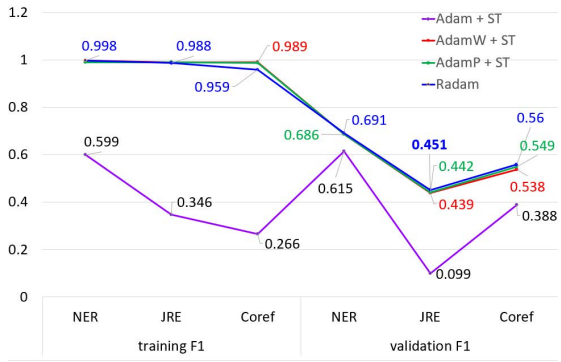

FIGURE 4. IE multi-task performance comparison according to optimizer; PLM is DeBERTa [10].

D. COMPARISON OF INFORMATION EXTRACTION PERFORMANCE CHANGES ACCORDING TO OPTIMIZERS

We compared the performance of four types of optimizers in the state of applying re-initialization to the 10th layer of DeBERTa PLM by the DyGIE++ model, which performs multi-task learning for 4 types of optimizers including Adam during fine-tuning. All experiments were also performed five times and the average value was taken in consideration of variance due to random initialization. Figure 4 shows performance of NER, JRE, and Coref for the SciERC dataset. The weights for multi-task learning of NER, RE, and Coref are 0.2, 1.0, and 1.0, respectively. For the target task, we used the 150-dimension dropout 0.4 for two layers. Table 3 shows the other hyperparameters. Figure 4 shows that RAdam without a scheduler achieve the best performance compared to other optimizers. In Figure 4, Adam, AdamW, and AdamP except for RAdam used the slanted triangular scheduler, and after RAdam, AdamP shows good performance. However, as previously mentioned in IV-C, AdamP has the disadvantage of slowing down the learning rate by approximately 8%. For AdamW, when the polynomial decay learning scheduler was applied to warm-up five epochs instead of slanted triangular scheduler, the same performance as AdamP was confirmed. As described in the Figure 4, the performance of RAdam is higher than Adam, AdamW and AdamP in all three kinds of tasks, NER, JRE and Coref.

VI. CONCLUSION

This study proposed SciDeBERTa through training specialized in the science domain; the base model was DeBERTa whose performance as a general LM benchmark has been proven. It was experimentally verified that SciDeBERTa and SciDeBERTa(CS) have improved performance compared to SciBERT and S2ORC-SciBERT, which are suggested as conventional LMs specialized in the science technology domain. In particular, in the case of SciDeBERTa (CS) trained in the CS domain among the science and technology domains, improved performance was confirmed compared to SciDeBERTa trained in various science and technology domains on the SciERC dataset. Furthermore, Xavier normalization was used to re-initialize the PLM, and the RAdam optimizer was applied as a method for overfitting resolution and optimization to improve LM performance by fine-tuning the PLM. For related research in the future, further tagging for the SciERC dataset or data augmentation is necessary, considering that the tagging of the SciERC dataset in the present model is insufficient.

ACKNOWLEDGMENT

This work was helped by the National Supercomputing Center with their resources and technical supports. *(All the authors contributed equally to this work.)*

REFERENCES

[1] T. Henighan, J. Kaplan, M. Katz, M. Chen, C. Hesse, J. Jackson, H. Jun, T. B. Brown, P. Dhariwal, S. Gray, C. Hallacy, B. Mann, A. Radford, A. Ramesh, N. Ryder, D. M. Ziegler, J. Schulman, D. Amodei, and S. McCandlish, "Scaling laws for autoregressive generative modeling," 2020, *arXiv:2010.14701*.

[2] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-LM: Training multi-billion parameter language models using model parallelism," 2019, *arXiv:1909.08053*.

[3] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, and S. Young, "Scaling language models: Methods, analysis & insights from training gopher," 2021, *arXiv:2112.11446*.

[4] Y. Sun, S. Wang, S. Feng, S. Ding, C. Pang, J. Shang, J. Liu, X. Chen, Y. Zhao, and Y. Lu, "ERNIE 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation," 2021, *arXiv:2107.02137*.

[5] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, and S. Gehrmann, "PaLM: Scaling language modeling with pathways," 2022, *arXiv:2204.02311*.

[6] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperGlue: Learning feature matching with graph neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 4938–4947.

[7] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," 2021, *arXiv:2101.03961*.
[8] Y. Meng, C. Xiong, P. Bajaj, S. Tiwary, P. Bennett, J. Han, and X. Song, "COCO-LM: Correcting and contrasting text sequences for language model pretraining," 2021, *arXiv:2102.08473*.
[9] Z. Wang, A. W. Yu, O. Firat, and Y. Cao, "Towards zero-label language learning," 2021, *arXiv:2109.09193*.
[10] P. He, X. Liu, J. Gao, and W. Chen, "DeBERTa: Decoding-enhanced BERT with disentangled attention," in Proc. 9th Int. Conf. Learn. Represent. (ICLR), 2021.
[11] I. Beltagy, K. Lo, and A. Cohan, "SciBERT: A pretrained language model for scientific text," in *Proc. Conf. Empirical Methods Natural Lang. Process., 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 3615–3620.
[12] K. Lo, L. L. Wang, M. Neumann, R. Kinney, and D. Weld, "S2ORC: The semantic scholar open research corpus," in *Proc. 58th Annu. Meeting Assoc. Comput. Linguistics*, 2020, pp. 4969–4983.
[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019, pp. 4171–4186
[14] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in*Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, 2019.
[15] B. Heo, S. Chun, S. J. Oh, D. Han, S. Yun, G. Kim, Y. Uh, and J. W. Ha, "AdamP: Slowing down the slowdown for momentum optimizers on scale-invariant weights," in *Proc. 9th Int. Conf. Learn. Represent. (ICLR)*, 2021.
[16] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the Variance of the Adaptive Learning Rate and Beyond," in *Proc. 8th Int. Conf. Learn. Represent. (ICLR)*, 2020.
[17] D. P. Kingma, J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015.
[18] 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. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.
[19] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. (2018). *Improving Language Understanding by Generative Pre-Training*. [Online]. Available: https://s3-us-west-2.amazonaws.com/openaiassets/research-covers/languageunsupervised/languageunderstandingpaper.pdf
[20] M. Joshi, D. Chen, Y. Liu, D. S. Weld, L. Zettlemoyer, and O. Levy, "SpanBERT: Improving pre-training by representing and predicting spans," *Trans. Assoc. Comput. Linguistics*, vol. 8, pp. 64–77, Jan. 2020.
[21] W. Wang, B. Bi, M. Yan, C. Wu, J. Xia, Z. Bao, L. Peng, and L. Si, "StructBERT: Incorporating language structures into pre-training for deep language understanding," in *Proc. 7th Int. Conf. Lear. Represen. (ICLR)*, 2019.
[22] Y. Sun, S. Wang, Y. Li, S. Feng, H. Tian, H. Wu, and H. Wang, "ERNIE 2.0: A continual pre-training framework for language understanding," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 5, pp. 8968–8975.
[23] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 19–27.
[24] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 5753–5763.
[25] M. M. A. Qudar and V. Mago, "TweetBERT: A pretrained language representation model for Twitter text analysis," 2020, *arXiv:2010.11091*.
[26] D. Araci, "FinBERT: Financial sentiment analysis with pre-trained language models," 2019, *arXiv:1908.10063*.
[27] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos, "LEGAL-BERT: The muppets straight out of law school," in *Proc. Findings Assoc. Comput. Linguistics*, 2020, pp. 2898–2904.
[28] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "BioBERT: A pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, p. 1234, Sep. 2019.
[29] Y. Gu, R. Tinn, H. Cheng, M. Lucas, N. Usuyama, X. Liu, T. Naumann, J. Gao, and H. Poon, "Domain-specific language model pretraining for biomedical natural language processing," *ACM Trans. Comput. Healthcare*, vol. 3, no. 1, pp. 1–23, Jan. 2022.
[30] T. Zhang, F. Wu, A. Katiyar, K. Q. Weinberger, and Y. Artzi, "Revisiting few-sample BERT fine-tuning," in *Proc. 9th Int. Conf. Learn. Represent. (ICLR)*, 2021.
[31] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu, "On layer normalization in the transformer architecture," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 10524–10533.
[32] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," 2021, *arXiv:2106.04554*.
[33] M. Mosbach, M. Andriushchenko, and D. Klakow, "On the stability of fine-tuning BERT: Misconceptions, explanations, and strong baselines," in *Proc. 9th Int. Conf. Learn. Represent. (ICLR)*, 2021.
[34] L. Liu, X. Liu, J. Gao, W. Chen, and J. Han, "Understanding the difficulty of training transformers," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2020, pp. 5747–5763.
[35] Y. Luan, L. He, M. Ostendorf, and H. Hajishirzi, "Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2018, pp. 3219–3232.
[36] J.-D. Kim, T. Ohta, Y. Tateisi, and J. Tsujii, "GENIA corpus—A semantically annotated corpus for bio-textmining," *Bioinformatics*, vol. 19, no. 1, pp. i180–i182, Jul. 2003.
[37] D. Wadden, U. Wennberg, Y. Luan, and H. Hajishirzi, "Entity, relation, and event extraction with contextualized span representations," in *Proc. Conf. Empirical Methods Natural Lang. Process., 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 5784–5789.
[38] B. van Aken, B. Winter, A. Löser, and F. A. Gers, "How does BERT answer questions?: A layer-wise analysis of transformer representations," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 1823–1832.
[39] K. Clark, U. Khandelwal, O. Levy, and C. D. Manning, "What does BERT look at? An analysis of BERT's attention," in *Proc. ACL Workshop BlackboxNLP, Analyzing Interpreting Neural Netw.*, 2019, pp. 276–286.
[40] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.
[41] B. QasemiZadeh and A.-K. Schumann, "The ACL RD-TEC 2.0: A language resource for evaluating term extraction and entity recognition methods," in *Proc. 10th Int. Conf. Lang. Resour. Eval. (LREC)*, 2016, pp. 1862–1868.

YUNA JEONG (Member, IEEE) received the B.S. degree in computer engineering from Korea Polytechnic University, in 2012, and the Ph.D. degree in computer engineering from Sungkyunkwan University, in 2019. She is currently a Senior Researcher with the AI Technology Research Center, Korea Institute of Science and Technology Information (KISTI). Her main research interests include machine learning, deep learning, and natural language processing.

EUNHUI KIM (Member, IEEE) received the B.S. degree in information communication engineering from Chungnam National University, South Korea, in 2000, and the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), South Korea, in 2009 and 2015, respectively. From 2000 to 2007, she was a Researcher with Samsung Electronics, Seoul, South Korea. From 2015 to 2017, she was a Postdoctoral Researcher with KAIST. In 2018, she was an Invited Professor with the National Center of Excellence in Software, Chungnam National University. Since 2019, she has been working with the AI Technology Center, Korea Institute of Science and Technology Information (KISTI), as a Senior Researcher. Her research interests include machine learning, recommendation systems, intelligent transportation, and lightweight deep neural networks modeling in vision and language processing.