Self-supervised Regularization for Text Classification

Meng Zhou∗
Shanghai Jiao Tong University, China
zhoumeng9904@sjtu.edu.cn

Zechen Li∗
Northeastern University, United States
li.zec@northeastern.edu

Pengtao Xie†
UC San Diego, United States
p1xie@eng.ucsd.edu

Abstract
Text classification is a widely studied problem and has broad applications. In many real-world problems, the number of texts for training classification models is limited, which renders these models prone to overfitting. To address this problem, we propose SSL-Reg, a data-dependent regularization approach based on self-supervised learning (SSL). SSL (Devlin et al., 2019a) is an unsupervised learning approach that defines auxiliary tasks on input data without using any human-provided labels and learns data representations by solving these auxiliary tasks. In SSL-Reg, a supervised classification task and an unsupervised SSL task are performed simultaneously. The SSL task is unsupervised, which is defined purely on input texts without using any human-provided labels. Training a model using an SSL task can prevent the model from being overfitted to a limited number of class labels in the classification task. Experiments on 17 text classification datasets demonstrate the effectiveness of our proposed method. Code is available at https://github.com/UCSD-AI4H/SSReg.

1 Introduction
Text classification (Korde and Mahender, 2012; Lai et al., 2015; Wang et al., 2017; Howard and Ruder, 2018) is a widely studied problem in natural language processing and finds broad applications. For example, given clinical notes of a patient, judge whether this patient has heart diseases. Given a scientific paper, judge whether it is about NLP. In many real-world text classification problems, texts available for training are oftentimes limited. For instance, it is difficult to obtain many clinical notes from hospitals due to concern of patient privacy. It is well known that when training data is limited, models tend to overfit to training data and perform less well on test data.

To address overfitting problems in text classification, we propose a data-dependent regularizer called SSL-Reg based on self-supervised learning (SSL) (Devlin et al., 2019a; He et al., 2019; Chen et al., 2020) and use it to regularize the training of text classification models, where a supervised classification task and an unsupervised SSL task are performed simultaneously. SSL (Devlin et al., 2019a; He et al., 2019; Chen et al., 2020) is an unsupervised learning approach that defines auxiliary tasks on input data without using any human-provided labels and learns data representations by solving these auxiliary tasks. For example, BERT (Devlin et al., 2019a) is a typical SSL approach where an auxiliary task is defined to predict masked tokens and a text encoder is learned by solving this task. In existing SSL approaches for NLP, an SSL task and a target task are performed sequentially. A text encoder is first trained by solving an SSL task defined on a large collection of unlabeled texts. Then this encoder is used to initialize an encoder in a target task. The encoder is finetuned by solving the target task. A potential drawback of performing SSL task and target task sequentially is that text encoder learned in SSL task may be overridden after being finetuned in target task. If training data in the target task is small, the finetuned encoder has a high risk of being overfitted to training data.

To address this problem, in SSL-Reg we perform SSL task and target tasks (which is classification) simultaneously. In SSL-Reg, an SSL loss serves as a regularization term and is optimized jointly with a classification loss. SSL-Reg enforces a text encoder to jointly solve two tasks: an unsupervised SSL task and a supervised text classification task. Due to the presence of the SSL task, models are less likely to be biased to the classification task defined on small-sized training
data. We perform experiments on 17 datasets, where experimental results demonstrate the effectiveness of SSL-Reg in alleviating overfitting and improving generalization performance.

The major contributions of this paper are:

- We propose SSL-Reg, which is a data-dependent regularizer based on SSL, to reduce the risk that a text encoder is biased to a data-deficient classification task on small-sized training data.
- Experiments on 17 datasets demonstrate the effectiveness of our approaches.

The rest of this paper is organized as follows. Section 2 reviews related works. Section 3 and 4 present methods and experiments. Section 5 concludes the paper and discusses future work.

2 Related Works

2.1 Self-supervised Learning for NLP

SSL aims to learn meaningful representations of input data without using human annotations. It creates auxiliary tasks solely using input data and forces deep networks to learn highly effective latent features by solving these auxiliary tasks. In NLP, various auxiliary tasks have been proposed for SSL, such as next token prediction in GPT (Radford et al.), masked token prediction in BERT (Devlin et al., 2019a), text denoising in BART (Lewis et al., 2019), contrastive learning (Fang et al., 2020), and so on. These models have achieved substantial success in learning language representations. The GPT model (Radford et al.) is a language model based on Transformer (Vaswani et al., 2017). Different from Transformer, which defines a conditional probability on an output sequence given an input sequence, GPT defines a marginal probability on a single sequence. In GPT, conditional probability of the next token given a historical sequence is defined using a Transformer decoder. Weight parameters are learned by maximizing likelihood on token sequences. BERT (Devlin et al., 2019a) aims to learn a Transformer encoder for representing texts. BERT's model architecture is a multi-layer bidirectional Transformer encoder. In BERT, Transformer uses bidirectional self-attention. To train the encoder, BERT masks some percentage of input tokens at random, and then predicts those masked tokens by feeding hidden vectors (produced by the encoder) corresponding to masked tokens into an output softmax over word vocabulary. BERT-GPT (Wu et al., 2019) is a model used for sequence-to-sequence modeling where a pretrained BERT is used to encode input text and GPT is used to generate output texts. In BERT-GPT, pretraining of BERT encoder and GPT decoder is conducted separately, which may lead to inferior performance. Auto-Regressive Transformers (BART) (Lewis et al., 2019) has a similar architecture as BERT-GPT, but trains BERT encoder and GPT decoder jointly. To pretrain BART weights, input texts are corrupted randomly, such as token masking, token deletion, text infilling, and so forth, then a network is learned to reconstruct original texts. ALBERT (Lan et al., 2019) uses parameter-reduction methods to reduce memory consumption and increase training speed of BERT. It also introduces a self-supervised loss that models inter-sentence coherence. RoBERTa (Liu et al., 2019a) is a replication study of BERT pretraining. It shows that BERT's performance can be greatly improved by carefully tuning training processes, such as (1) training models longer, with larger batches, over more data; (2) removing the next sentence prediction objective; (3) training on longer sequences, and so on. XLNet (Yang et al., 2019) learns bidirectional contexts by maximizing expected likelihood over all permutations of factorization order and uses a generalized autoregressive pretraining mechanism to overcome the pretrain-finetune discrepancy of BERT. T5 (Raffel et al., 2019) compared pretraining objectives, architectures, unlabeled datasets, transfer approaches on a wide range of language understanding tasks and proposed a unified framework that casts these tasks as a text-to-text task. ERNIE 2.0 (Sun et al., 2019) proposed a continual pretraining framework which builds and learns incrementally pretraining tasks through constant multi-task learning, to capture lexical, syntactic and semantic information from training corpora. Gururangan et al. (2020) proposed task adaptive pretraining (TAPT) and domain adaptive pretraining (DAPT). Given a RoBERTa model pretrained on large-scale corpora, TAPT continues to pretrain RoBERTa on training dataset of target task. DAPT continues to pretrain RoBERTa on datasets that have small domain differences with data in target tasks. The difference between our proposed SSL-Reg method with TAPT and DAPT is that SSL-Reg uses a self-supervised task (e.g., mask token prediction) to regularize the finetuning of
RoBERTa where text classification task and self-supervised task are performed jointly. In contrast, TAPT and DAPT use self-supervised task for pretraining, where text classification task and self-supervised task are performed sequentially. The connection between our method and TAPT is that they both leverage texts in target tasks to perform self-supervised learning, in addition to SSL on large-scale external corpora. Different from SSL-Reg and TAPT, DAPT uses domain-similar texts rather than target texts for additional SSL.

2.2 Self-supervised Learning in General

Self-supervised learning has been widely applied to other application domains, such as image classification (He et al., 2019; Chen et al., 2020), graph classification (Zeng and Xie, 2021), visual question answering (He et al., 2020a), and so forth, where various strategies have been proposed to construct auxiliary tasks, based on temporal correspondence (Li et al., 2019; Wang et al., 2019a), cross-modal consistency (Wang et al., 2019b), rotation prediction (Gidaris et al., 2018; Sun et al., 2020), image inpainting (Pathak et al., 2016), automatic colorization (Zhang et al., 2016), context prediction (Nathan Mundhenk et al., 2018), and so on. Some recent works studied self-supervised representation learning based on instance discrimination (Wu et al., 2018) with contrastive learning. Oord et al. (2018) proposed contrastive predictive coding, which predicts the future in latent space by using powerful autoregressive models, to extract useful representations from high-dimensional data. Bachman et al. (2019) proposed a self-supervised representation learning approach based on maximizing mutual information between features extracted from multiple views of a shared context. MoCo (He et al., 2019) and SimCLR (Chen et al., 2020) learned image encoders by predicting whether two augmented images were created from the same original image. Srinivas et al. (2020) proposed a contrastive unsupervised approach which applied contrastive self-supervised learning on top of networks pretrained by transfer learning.

Compared with supervised learning that requires each data example to be labeled by humans or semi-supervised learning which requires part of data examples to be labeled, self-supervised learning is similar to unsupervised learning because it does not need human-provided labels. The key difference between SSL and unsupervised learning is that SSL focuses on learning data representations by solving auxiliary tasks defined on un-labeled data while unsupervised learning is more general and aims to discover latent structures from data, such as clustering, dimension reduction, manifold embedding (Roweis and Saul, 2000), and so on.

2.3 Text Classification

Text classification (Minaee et al., 2020) is one of the key tasks in natural language processing and has a wide range of applications, such as sentiment analysis, spam detection, tag suggestion, and so forth. A number of approaches have been proposed for text classification. Many of them are based on RNNs. Liu et al. (2016) use multi-task learning to train RNNs, utilizing the correlation between tasks to improve text classification performance. Tai et al. (2015) generalize sequential LSTM to tree-structured LSTM to capture the syntax of sentences for achieving better classification performance. Compared with RNN-based models, CNN-based models are good at capturing local and position-invariant patterns. Kalchbrenner et al. (2014) proposed dynamic CNN, which uses dynamic k-max-pooling to explicitly capture short-term and long-range relations of words and phrases. Zhang et al. (2015) proposed a character-level CNN model for text classification, which can deal with out-of-vocabulary words. Hybrid methods combine RNN and CNN to explore the advantages of both. Zhou et al. (2015) proposed a convolutional LSTM network, which uses a CNN to extract phrase-level representations, then feeds them to an LSTM network to represent the whole sentence.

3 Methods

To alleviate overfitting in text classification, we propose SSL-Reg, which is a regularization
approach based on self-supervised learning (SSL), where an unsupervised SSL task and a supervised text classification task are performed jointly.

### 3.1 SSL-based Regularization

SSL-Reg uses a self-supervised learning task to regularize a text classification model. Figure 1 presents an illustration of SSL-Reg. Given training texts, we encode them using a text encoder. Then on top of text encodings, two tasks are defined. One is a classification task, which takes the encoding of a text as input and predicts the class label of this text. Prediction is conducted using a classification head. The other task is SSL. The loss of the SSL task serves as a data-dependent regularizer to alleviate overfitting. The SSL task has a predictive head. These two tasks share the same text encoder. Formally, SSL-Reg solves the following optimization problem:

$$\mathcal{L}(c)(D, L; W^{(e)}, W^{(c)}) + \lambda \mathcal{L}(p)(D, W^{(e)}, W^{(p)})$$

(1)

where $D$ represents training texts and $L$ represents their labels. $W^{(e)}, W^{(c)},$ and $W^{(p)}$ denote text encoder, classification head in classification task, and prediction head in SSL task, respectively. $\mathcal{L}(c)$ denotes classification loss and $\mathcal{L}(p)$ denotes SSL loss. $\lambda$ is a tradeoff parameter.

At the core of SSL-Reg is using SSL to learn a text encoder that is robust to overfitting. Our methods can be used to learn any text encoder. In this work, we perform the study using a Transformer encoder, while noting that other text encoders are also applicable.

### 3.2 Self-supervised Learning Tasks

In this work, we use two self-supervised learning tasks—masked token prediction (MTP) and sentence augmentation type prediction (SATP)—to perform our studies while noting that other SSL tasks are also applicable.

- **Masked Token Prediction (MTP)** This task is used in BERT. Some percentage of input tokens are masked at random. Texts with masked tokens are fed into a text encoder that learns a latent representation for each token including the masked ones. The task is to predict these masked tokens by feeding hidden vectors (produced by the encoder) corresponding to masked tokens into an output softmax over word vocabulary.

- **Sentence Augmentation Type Prediction (SATP)** Given an original text $o$, we apply different types of augmentation methods to create augmented texts from $o$. We train a model to predict which type of augmentation was applied to an augmented text. We consider four types of augmentation operations used in Wei and Zou (2019), including synonym replacement, random insertion, random swap, and random deletion. Synonym replacement randomly chooses 10% of non-stop tokens from original texts and replaces each of them with a randomly selected synonym. In random insertion, for a randomly chosen non-stop token in a text, among the synonyms of this token, one randomly selected synonym is inserted into a random position in the text. This operation is performed for 10% of tokens. Synonyms for synonym replacement and random insertion are obtained from Synsets in NLTK (Bird and Loper, 2004) which are constructed based on WordNet (Miller, 1995). Synsets serve as a synonym dictionary containing groupings of synonymous words. Some words have only one Synset and some have several. In synonym replacement, if a selected word in a sentence has multiple synonyms, we randomly choose one of them, and replace all occurrences of this word in the sentence.
with this synonym. Random swap randomly chooses two tokens in a text and swaps their positions. This operation is performed for 10% of token pairs. Random deletion randomly removes a token with a probability of 0.1. In this SSL task, an augmented sentence is fed into a text encoder and the encoding is fed into a 4-way classification head to predict which operation was applied to generate this augmented sentence.

### 3.3 Text Encoder

We use a Transformer encoder to perform the study while noting that other text encoders are also applicable. Different from sequence-to-sequence models (Sutskever et al., 2014) that are based on recurrent neural networks (e.g., LSTM [Hochreiter and Schmidhuber, 1997], GRU [Chung et al., 2014]), which model a sequence of tokens via a recurrent manner and hence are computationally inefficient, Transformer eschews recurrent computation and instead uses self-attention which not only can capture dependency between tokens but also is amenable for parallel computation with high efficiency. Self-attention calculates the correlation among every pair of tokens and uses these correlation scores to create “attentive” representations by taking weighted summation of tokens’ embeddings. Transformer is composed of building blocks, each consisting of a self-attention layer and a position-wise feed-forward layer. Residual connection (He et al., 2016) is applied around each of these two sub-layers, followed by layer normalization (Ba et al., 2016). Given an input sequence, an encoder—which is a stack of such building blocks—is applied to obtain a representation for each token.

### 4 Experiments

#### 4.1 Datasets

We evaluated our method on the datasets used in Gururangan et al. (2020), which are from various domains. For each dataset, we follow the train/development/test split specified in Gururangan et al. (2020). Dataset statistics are summarized in Table 1.

In addition, we performed experiments on the datasets in the GLUE benchmark (Wang et al., 2018). The General Language Understanding Evaluation (GLUE) benchmark has 10 tasks, including 2 single-sentence tasks, 3 similarity and
paraphrase tasks, and 5 inference tasks. For each GLUE task, labels in development sets are publicly available and those in test sets are not released. We obtain performance on test sets by submitting inference results to GLUE evaluation server. Table 2 shows the statistics of data split in each task.

4.2 Experimental Setup

4.2.1 Baselines

For experiments on datasets used in Gururangan et al. (2020), text encoders in all methods are initialized using pretrained RoBERTa (Liu et al., 2019a). For experiments on GLUE datasets, text encoders are initialized using pretrained BERT (Liu et al., 2019a) or pretrained RoBERTa. We compare our proposed SSL-Reg with the following baselines.

- **Unregularized RoBERTa** (Liu et al., 2019b). In this approach, the Transformer encoder is initialized with pretrained RoBERTa. Then the pretrained encoder and a classification head form a text classification model, which is then finetuned on a target classification task. Architecture of the classification model is the same as that in Liu et al. (2019b). Specifically, representation of the [CLS] special token is passed to a feedforward layer for class prediction. Nonlinear activation function in the feedforward layer is tanh. During finetuning, no SSL-based regularization is used. This approach is evaluated on all datasets used in Gururangan et al. (2020) and all datasets in GLUE.

- **Unregularized BERT.** This approach is the same as unregularized RoBERTa, except that the Transformer encoder is initialized by pretrained BERT (Devlin et al., 2019a) instead of RoBERTa. This approach is evaluated on all GLUE datasets.

- **Task adaptive pretraining (TAPT)** (Gururangan et al., 2020). In this approach, given the Transformer encoder pretrained using RoBERTa or BERT on large-scale external corpora, it is further pretrained by RoBERTa or BERT on input texts in a target classification dataset (without using class labels). Then this further pretrained encoder is used to initialize the encoder in the text classification model and is finetuned to perform classification tasks which use both input texts and their class labels. Similar to SSL-Reg, TAPT also performs SSL on texts in target classification dataset. The difference is: TAPT performs SSL task and classification task sequentially while SSL-Reg performs these two tasks jointly. TAPT is studied for all datasets in this paper.

- **Domain adaptive pretraining (DAPT)** (Gururangan et al., 2020). In this approach, given a pretrained encoder on large-scale external corpora, the encoder is further pretrained on a small-scale corpora whose domain is similar to that of texts in a target classification dataset. Then this further pretrained encoder is finetuned in a classification task. DAPT is similar to TAPT, except that TAPT performs the second stage pretraining on texts T in the classification dataset while DAPT performs the second stage pretraining on external texts whose domain is similar to that of T rather than directly on T. The external dataset is usually much larger than T.

- **TAPT+SSL-Reg.** When finetuning the classification model, SSL-Reg is applied. The rest is the same as TAPT.

- **DAPT+SSL-Reg.** When finetuning the classification model, SSL-Reg is applied. The rest is the same as DAPT.

4.2.2 Hyperparameter Settings

Hyperparameters were tuned on development datasets.

Hyperparameter settings for RoBERTa on datasets used in Gururangan et al. (2020). For a fair comparison, most of our hyperparameters are the same as those in Gururangan et al. (2020). The maximum text length was set to 512. Text encoders in all methods are initialized using pretrained RoBERTa (Liu et al., 2019a) on a large-scale external dataset. For TAPT, DAPT, TAPT+SSL-Reg, and DAPT+SSL-Reg, the second-stage pretraining on texts T in a target classification dataset or on external texts whose domain is similar to that of T is based on the pretraining approach in RoBERTa. In SSL-Reg, the SSL task is masked token prediction. SSL loss function only considers

---

4https://gluebenchmark.com/leaderboard
Table 3: Hyperparameter settings for BERT on GLUE datasets, where the SSL task is MTP.

<table>
<thead>
<tr>
<th>Task</th>
<th>Epoch</th>
<th>Learning Rate</th>
<th>Regularization Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoLA</td>
<td>10</td>
<td>3e-5</td>
<td>0.2</td>
</tr>
<tr>
<td>SST-2</td>
<td>3</td>
<td>3e-5</td>
<td>0.05</td>
</tr>
<tr>
<td>MRPC</td>
<td>5</td>
<td>4e-5</td>
<td>0.05</td>
</tr>
<tr>
<td>STS-B</td>
<td>10</td>
<td>4e-5</td>
<td>0.1</td>
</tr>
<tr>
<td>QQP</td>
<td>5</td>
<td>3e-5</td>
<td>0.2</td>
</tr>
<tr>
<td>MNLI</td>
<td>3</td>
<td>3e-5</td>
<td>0.1</td>
</tr>
<tr>
<td>QNLI</td>
<td>4</td>
<td>4e-5</td>
<td>0.5</td>
</tr>
<tr>
<td>RTE</td>
<td>10</td>
<td>3e-5</td>
<td>0.1</td>
</tr>
<tr>
<td>WNLI</td>
<td>5</td>
<td>5e-5</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 4: Hyperparameter settings for BERT on GLUE datasets, where the SSL task is SATP.

<table>
<thead>
<tr>
<th>Task</th>
<th>Epoch</th>
<th>Learning Rate</th>
<th>Regularization Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoLA</td>
<td>6</td>
<td>3e-5</td>
<td>0.4</td>
</tr>
<tr>
<td>SST-2</td>
<td>3</td>
<td>3e-5</td>
<td>0.8</td>
</tr>
<tr>
<td>MRPC</td>
<td>5</td>
<td>4e-5</td>
<td>0.05</td>
</tr>
<tr>
<td>STS-B</td>
<td>10</td>
<td>4e-5</td>
<td>0.05</td>
</tr>
<tr>
<td>QQP</td>
<td>5</td>
<td>3e-5</td>
<td>0.4</td>
</tr>
<tr>
<td>MNLI</td>
<td>4</td>
<td>3e-5</td>
<td>0.5</td>
</tr>
<tr>
<td>QNLI</td>
<td>4</td>
<td>4e-5</td>
<td>0.05</td>
</tr>
<tr>
<td>RTE</td>
<td>8</td>
<td>3e-5</td>
<td>0.6</td>
</tr>
<tr>
<td>WNLI</td>
<td>5</td>
<td>5e-5</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 5: Hyperparameter settings for RoBERTa on GLUE datasets, where the SSL task is MTP.

<table>
<thead>
<tr>
<th>Task</th>
<th>Epoch</th>
<th>Learning Rate</th>
<th>Regularization Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoLA</td>
<td>10</td>
<td>1e-5</td>
<td>0.8</td>
</tr>
<tr>
<td>SST-2</td>
<td>3</td>
<td>1e-5</td>
<td>1.0</td>
</tr>
<tr>
<td>MRPC</td>
<td>10</td>
<td>1e-5</td>
<td>0.01</td>
</tr>
<tr>
<td>STS-B</td>
<td>10</td>
<td>2e-5</td>
<td>0.01</td>
</tr>
<tr>
<td>QQP</td>
<td>10</td>
<td>1e-5</td>
<td>0.1</td>
</tr>
<tr>
<td>MNLI</td>
<td>3</td>
<td>1e-5</td>
<td>0.1</td>
</tr>
<tr>
<td>QNLI</td>
<td>3</td>
<td>1e-5</td>
<td>0.1</td>
</tr>
<tr>
<td>RTE</td>
<td>10</td>
<td>2e-5</td>
<td>0.1</td>
</tr>
<tr>
<td>WNLI</td>
<td>10</td>
<td>2e-5</td>
<td>0.02</td>
</tr>
</tbody>
</table>

Hyperparameter settings for BERT on GLUE datasets. The maximum text length was set to 128. Since external texts whose domains are similar to those of the GLUE texts are not available, we did not compare with DAPT and DAPT+SSL-Reg. For each method applied, text encoder is initialized using pretrained BERT (Devlin et al., 2019a) (with 24 layers) on a large-scale external dataset. In TAPT, the second-stage pretraining is performed using BERT. As we will show later on, TAPT does not perform well on GLUE datasets; therefore, we did not apply TAPT+SSL-Reg on these datasets further. In SSL-Reg, we studied two SSL tasks: masked token prediction (MTP) and sentence augmentation type prediction (SATP). In MTP, we randomly mask 15% of tokens in each text. Batch size was set to 32 with gradient accumulation. We use the AdamW optimizer (Loshchilov and Hutter, 2017) with a warm-up proportion of 0.1, a weight decay of 0.1, and an epsilon of 1e-6. In AdamW, $\beta_1$ and $\beta_2$ are set to 0.9 and 0.98, respectively. The maximum learning rate was 2e-5.

Hyperparameter settings for RoBERTa on GLUE datasets. Most hyperparameter settings follow those in RoBERTa experiments performed on datasets used in Gururangan et al. (2020). We set different learning rates and different epoch numbers for different datasets as guided by Liu et al. (2019b). In addition, we set different regularization parameters for different datasets. These hyperparameters are listed in Table 5.
4.3 Results

4.3.1 Results on the Datasets used in Gururangan et al. (2020)

Performance of text classification on datasets used in Gururangan et al. (2020) is reported in Table 6. Following Gururangan et al. (2020), for CHEMPROT and RCT, we report micro-F1; for other datasets, we report macro-F1. DAPT, and TAPT, results are taken from Gururangan et al. (2020). For each method on each dataset, we run it for four times with different random seeds. Results are in \( m_s \) format, where \( m \) denotes mean and \( s \) denotes standard derivation. Following Gururangan et al. (2020), for CHEMPROT and RCT, we report micro-F1; for other datasets, we report macro-F1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>RoBERTa</th>
<th>DAPT</th>
<th>TAPT</th>
<th>SSL-Reg</th>
<th>TAPT+SSL-Reg</th>
<th>DAPT+SSL-Reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHEMPROT</td>
<td>81.9(_{0.1})</td>
<td>84.2(_{0.2})</td>
<td>82.6(_{0.4})</td>
<td>83.1(_{0.5})</td>
<td>83.5(_{0.1})</td>
<td>84.4(_{0.3})</td>
</tr>
<tr>
<td>RCT</td>
<td>87.2(_{0.1})</td>
<td>87.6(_{0.1})</td>
<td>87.7(_{0.1})</td>
<td>87.4(_{0.1})</td>
<td>87.7(_{0.1})</td>
<td>87.7(_{0.1})</td>
</tr>
<tr>
<td>ACL-ARC</td>
<td>63.0(_{0.8})</td>
<td>75.4(_{2.5})</td>
<td>67.4(_{1.8})</td>
<td>69.3(_{4.9})</td>
<td>68.1(_{2.0})</td>
<td>75.7(_{1.4})</td>
</tr>
<tr>
<td>SciERC</td>
<td>77.3(_{1.9})</td>
<td>80.8(_{1.5})</td>
<td>79.3(_{1.5})</td>
<td>81.4(_{0.8})</td>
<td>80.4(_{0.6})</td>
<td>82.3(_{0.8})</td>
</tr>
<tr>
<td>HYPERPARTISAN</td>
<td>86.6(_{0.9})</td>
<td>88.2(_{5.9})</td>
<td>90.4(_{5.2})</td>
<td>92.3(_{1.4})</td>
<td>93.2(_{1.8})</td>
<td>90.7(_{3.2})</td>
</tr>
<tr>
<td>AGNEWS</td>
<td>93.9(_{0.2})</td>
<td>93.9(_{0.2})</td>
<td>94.5(_{0.1})</td>
<td>94.2(_{0.1})</td>
<td>94.4(_{0.1})</td>
<td>94.0(_{0.1})</td>
</tr>
<tr>
<td>HELPFULNESS</td>
<td>65.1(_{3.4})</td>
<td>66.5(_{1.4})</td>
<td>68.5(_{1.9})</td>
<td>69.4(_{0.2})</td>
<td>71.0(_{1.0})</td>
<td>68.3(_{1.4})</td>
</tr>
<tr>
<td>IMDB</td>
<td>95.0(_{0.2})</td>
<td>95.4(_{0.1})</td>
<td>95.5(_{0.1})</td>
<td>95.7(_{0.1})</td>
<td>96.1(_{0.1})</td>
<td>95.4(_{0.1})</td>
</tr>
</tbody>
</table>

Table 6: Results on datasets used in Gururangan et al. (2020). For vanilla (unregularized) RoBERTa, DAPT, and TAPT, results are taken from Gururangan et al. (2020). For each method on each dataset, we run it for four times with different random seeds. Results are in \( m_s \) format, where \( m \) denotes mean and \( s \) denotes standard derivation. Following Gururangan et al. (2020), for CHEMPROT and RCT, we report micro-F1; for other datasets, we report macro-F1.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>RoBERTa</th>
<th>SSL-Reg</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHEMPROT</td>
<td>13.05</td>
<td>13.57</td>
</tr>
<tr>
<td>ACL-ARC</td>
<td>28.67</td>
<td>25.24</td>
</tr>
<tr>
<td>SciERC</td>
<td>19.51</td>
<td>18.23</td>
</tr>
<tr>
<td>HYPERPARTISAN</td>
<td>7.44</td>
<td>5.64</td>
</tr>
</tbody>
</table>

Table 7: Difference between F1 score on training set and F1 score on test set with or without SSL-Reg (MTP). **Bold** denotes a smaller difference, which means overfitting is less severe.
classification tasks and SSL tasks simultaneously. Thus the encoder is not completely biased to classification tasks. In TAPT, the encoder is first learned by performing SSL tasks, then finetuned by performing classification tasks. There is a risk that after finetuning, the encoder is largely biased to classification tasks on small-sized training data, which leads to overfitting.

Third, on 5 out of the 8 datasets, SSL-Reg performs better than DAPT, although DAPT leverages additional external data. The reasons are two-fold: 1) similar to TAPT, DAPT performs SSL task first and then classification task separately; as a result, the encoder may be eventually biased to classification task on small-sized training data; 2) external data used in DAPT still has a domain shift with target dataset; this domain shift may render the text encoder pretrained on external data not suitable for target task. To verify this, we measure the domain similarity between external texts and target texts by calculating cosine similarity between the BERT embeddings of these texts. The similarity score is 0.14. As a reference, the similarity score between texts in the target dataset is 0.27. This shows that there is indeed a domain difference between external texts and target texts.

Fourth, on 6 out of 8 datasets, TAPT+SSL-Reg performs better than TAPT. On the other two datasets, TAPT+SSL-Reg is on par with TAPT. This further demonstrates the effectiveness of SSL-Reg.

Fifth, on all eight datasets, DAPT+SSL-Reg performs better than DAPT. This again shows that SSL-Reg is effective.

Sixth, on 6 out of 8 datasets, TAPT+SSL-Reg performs better than SSL-Reg, indicating that it is beneficial to use both TAPT and SSL-Reg: first use the target texts to pretrain the encoder based on SSL, then apply SSL-based regularizer on these target texts during finetuning.

Seventh, DAPT+SSL-Reg performs better than SSL-Reg on 4 datasets, but worse on the other 4 datasets, indicating that with SSL-Reg used, DAPT is not necessarily useful.

Eighth, on smaller datasets, improvement achieved by SSL-Reg over baselines is larger. For example, on HYPERPARTISAN which has only about 500 training examples, improvement of SSL-Reg over RoBERTa is 5.7% (absolute percentage). Relative improvement is 6.6%. As another example, on ACL-ARC which has about 1700 training examples, improvement of SSL-Reg over RoBERTa is 6.3% (absolute percentage).
<table>
<thead>
<tr>
<th></th>
<th>CoLA (Matthew Corr.)</th>
<th>SST-2 (Accuracy)</th>
<th>RTE (Accuracy)</th>
<th>QNLI (Accuracy)</th>
<th>MRPC (Accuracy/F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>The median result</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BERT, Lan et al. 2019</td>
<td>60.6</td>
<td>93.2</td>
<td>70.4</td>
<td>92.3</td>
<td><strong>88.0/-</strong></td>
</tr>
<tr>
<td>BERT, our run</td>
<td>62.1</td>
<td>93.1</td>
<td>74.0</td>
<td>92.1</td>
<td>86.8/90.8</td>
</tr>
<tr>
<td>TAPT</td>
<td>61.2</td>
<td>93.1</td>
<td>74.0</td>
<td>92.0</td>
<td>85.3/89.8</td>
</tr>
<tr>
<td>SSL-Reg (SATP)</td>
<td>63.7</td>
<td><strong>93.9</strong></td>
<td>74.7</td>
<td>92.3</td>
<td>86.5/90.3</td>
</tr>
<tr>
<td>SSL-Reg (MTP)</td>
<td><strong>63.8</strong></td>
<td>93.8</td>
<td><strong>74.7</strong></td>
<td><strong>92.6</strong></td>
<td>87.3/90.9</td>
</tr>
<tr>
<td><strong>The best result</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BERT, our run</td>
<td>63.9</td>
<td>93.3</td>
<td>75.8</td>
<td>92.5</td>
<td><strong>89.5/92.6</strong></td>
</tr>
<tr>
<td>TAPT</td>
<td>62.0</td>
<td>93.9</td>
<td>76.2</td>
<td>92.4</td>
<td>86.5/90.7</td>
</tr>
<tr>
<td>SSL-Reg (SATP)</td>
<td>65.3</td>
<td>94.6</td>
<td><strong>78.0</strong></td>
<td>92.8</td>
<td>88.5/91.9</td>
</tr>
<tr>
<td>SSL-Reg (MTP)</td>
<td><strong>66.3</strong></td>
<td><strong>94.7</strong></td>
<td><strong>78.0</strong></td>
<td><strong>93.1</strong></td>
<td><strong>89.5/92.4</strong></td>
</tr>
</tbody>
</table>

Table 8: Results of BERT-based experiments on GLUE development sets, where results on MNLI and QQP are the median of five runs and results on other datasets are the median of nine runs. The size of MNLI and QQP is very large, taking a long time to train on. Therefore, we reduced the number of runs. Because we used a different optimization method to re-implement BERT, our median performance is not the same as that reported in Lan et al. (2019).

Relative improvement is 10%. In contrast, on large datasets such as RCT which contains about 180000 training examples, improvement of SSL-Reg over RoBERTa is 0.2% (absolute percentage). Relative improvement is 0.2%. On another large dataset AGNews which contains 115000 training examples, improvement of SSL-Reg over RoBERTa is 0.3% (absolute percentage). Relative improvement is 0.3%. The reason that SSL-Reg achieves better improvement on smaller datasets is that smaller datasets are more likely to lead to overfitting and SSL-Reg is more needed to alleviate this overfitting.

Figure 3 shows how classification F1 score varies as we increase regularization parameter $\lambda$ from 0.01 to 1.0 in SSL-Reg. As can be seen, starting from 0.01, when the regularizer parameter is increasing, F1 score increases. This is because a larger $\lambda$ imposes a stronger regularization effect, which helps to reduce overfitting. However, if $\lambda$ becomes too large, F1 score drops. This is because the regularization effect is too strong, which dominates classification loss. Among these 4 datasets, F1 score drops dramatically on HYPERPARTISAN as $\lambda$ increases. This is probably because this dataset contains very long sequences. This makes MTP on this dataset more difficult and therefore yields an excessively strong regularization outcome that hurts classification performance. Compared with HYPERPARTISAN, F1 score is less sensitive on other datasets because their sequence lengths are relatively smaller.

### 4.3.2 Results on the GLUE Benchmark

Table 8 and Table 9 show results of BERT-based experiments on development sets of GLUE. As mentioned in (Devlin et al., 2019b), for the 24-layer version of BERT, finetuning is sometimes unstable on small datasets, so we run each method several times and report the median and best performance. Table 10 shows the best performance on test sets. Following Wang et al. (2018), we report Matthew correlation on CoLA, Pearson correlation and Spearman correlation on STS-B, accuracy and F1 on MRPC and QQP. For the rest of the datasets, we report accuracy. From these tables, we make the following observations. First, SSL-Reg methods including SSL-Reg-SATP and SSL-Reg-MTP outperform unregularized BERT (our run) on most datasets: 1) on test sets, SSL-Reg-SATP performs better than BERT on 7 out of 10 datasets and SSL-Reg-MTP performs better than BERT on 9 out of 10 datasets; 2) in terms of median results on development sets, SSL-Reg-SATP performs better than BERT (our run) on 7 out of 9 datasets and SSL-Reg-MTP performs better than BERT (our run) on 8 out of 9 datasets; 3) in terms of best results on development sets, SSL-Reg-SATP performs better than BERT (our run) on 8 out of 9 datasets and SSL-Reg-MTP performs better than...
Table 9: Continuation of Table 8.

<table>
<thead>
<tr>
<th></th>
<th>BERT</th>
<th>TAPT</th>
<th>SSL-Reg (SATP)</th>
<th>SSL-Reg (MTP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoLA (Matthew Corr.)</td>
<td>60.5</td>
<td>61.3</td>
<td><strong>63.0</strong></td>
<td>61.2</td>
</tr>
<tr>
<td>SST-2 (Accuracy)</td>
<td>94.9</td>
<td>94.4</td>
<td>95.1</td>
<td><strong>95.2</strong></td>
</tr>
<tr>
<td>RTE (Accuracy)</td>
<td>70.1</td>
<td>70.3</td>
<td>71.2</td>
<td><strong>72.7</strong></td>
</tr>
<tr>
<td>QNLI (Accuracy)</td>
<td>92.7</td>
<td>92.4</td>
<td>92.5</td>
<td><strong>93.2</strong></td>
</tr>
<tr>
<td>MRPC (Accuracy/F1)</td>
<td>85.4/89.3</td>
<td>85.9/89.5</td>
<td>85.3/89.3</td>
<td><strong>86.1/89.8</strong></td>
</tr>
<tr>
<td>MNLI-m/mm (Accuracy)</td>
<td><strong>86.7/85.9</strong></td>
<td>85.7/84.4</td>
<td>86.2/85.4</td>
<td><strong>86.6/86.1</strong></td>
</tr>
<tr>
<td>QQP (Accuracy/F1)</td>
<td>89.3/72.1</td>
<td>89.3/71.6</td>
<td>89.6/72.2</td>
<td><strong>89.7/72.5</strong></td>
</tr>
<tr>
<td>STS-B (Pearson Corr./Spearman Corr.)</td>
<td>87.6/86.5</td>
<td><strong>88.4/87.3</strong></td>
<td>88.3/87.5</td>
<td>88.1/87.2</td>
</tr>
<tr>
<td>WNLI (Accuracy)</td>
<td>65.1</td>
<td>65.8</td>
<td>65.8</td>
<td><strong>66.4</strong></td>
</tr>
<tr>
<td>AX(Matthew Corr.)</td>
<td>39.6</td>
<td>39.3</td>
<td>40.2</td>
<td><strong>40.3</strong></td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>80.5</strong></td>
<td><strong>80.6</strong></td>
<td><strong>81.0</strong></td>
<td><strong>81.3</strong></td>
</tr>
</tbody>
</table>

Table 10: Results of BERT-based experiments on GLUE test sets, which are scored by the GLUE evaluation server (https://gluebenchmark.com/leaderboard). Models evaluated on AX are trained on the training dataset of MNLI.

BERT (our run) on 8 out of 9 datasets. This further demonstrates the effectiveness of SSL-Reg in improving generalization performance.

Second, on 7 out of 10 test sets, SSL-Reg-SATP outperforms TAPT; on 8 out of 10 test sets, SSL-Reg-MTP outperforms TAPT. On most development datasets, SSL-Reg-SATP and SSL-Reg-MTP outperform TAPT. The only exception is: on QQP development set, the best F1 of TAPT is slightly better than that of SSL-Reg-SATP. This further demonstrates that performing SSL-based regularization on target texts is more effective than using them for pretraining.

Third, overall, SSL-Reg-MTP performs better than SSL-Reg-SATP. For example, on 8 out of 10 test datasets, SSL-Reg-MTP performs better than SSL-Reg-SATP. MTP works better than SATP probably because it is a more challenging self-supervised learning task that encourages encoders to learn more powerful representations.

Fourth, improvement of SSL-Reg methods over BERT is more prominent on smaller training datasets, such as CoLA and RTE. This may be because smaller training datasets are more likely to lead to overfitting where the advantage of SSL-Reg in alleviating overfitting can be better played.

Tables 11 and 12 show results of RoBERTa-based experiments on development sets of GLUE. From these two tables, we make observations that are similar to those in Table 8 and Table 9. In terms of median results, SSL-Reg (MTP) performs better than unregularized RoBERTa (our
Table 11: Results of RoBERTa-based experiments on GLUE development sets, where the median results are the median of five runs. Because we used a different optimization method to re-implement RoBERTa, our median performance is not the same as that reported in (Liu et al., 2019b).

<table>
<thead>
<tr>
<th></th>
<th>CoLA (Matthew Corr.)</th>
<th>SST-2 (Accuracy)</th>
<th>RTE (Accuracy)</th>
<th>QNLI (Accuracy)</th>
<th>MRPC (Accuracy/F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>The median result</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RoBERTa, Liu et al. 2019</td>
<td>68.0</td>
<td><strong>96.4</strong></td>
<td>86.6</td>
<td>94.7</td>
<td><strong>90.9/-</strong></td>
</tr>
<tr>
<td>RoBERTa, our run</td>
<td>68.7</td>
<td>96.1</td>
<td>84.8</td>
<td>94.6</td>
<td>89.5/92.3</td>
</tr>
<tr>
<td>SSL-Reg (MTP)</td>
<td><strong>69.2</strong></td>
<td><strong>96.3</strong></td>
<td><strong>85.2</strong></td>
<td><strong>94.9</strong></td>
<td>90.0/92.7</td>
</tr>
</tbody>
</table>

The best result

<table>
<thead>
<tr>
<th></th>
<th>CoLA (Matthew Corr.)</th>
<th>SST-2 (Accuracy)</th>
<th>RTE (Accuracy)</th>
<th>QNLI (Accuracy)</th>
<th>MRPC (Accuracy/F1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoBERTa, our run</td>
<td>69.2</td>
<td>96.7</td>
<td>86.6</td>
<td>94.7</td>
<td>90.4/93.1</td>
</tr>
<tr>
<td>SSL-Reg (MTP)</td>
<td><strong>70.2</strong></td>
<td><strong>96.7</strong></td>
<td><strong>86.6</strong></td>
<td><strong>95.2</strong></td>
<td><strong>91.4/93.8</strong></td>
</tr>
</tbody>
</table>

Table 12: Continuation of Table 11.

<table>
<thead>
<tr>
<th></th>
<th>MNLI-m/mm (Accuracy)</th>
<th>QQP (Accuracy)</th>
<th>STS-B (Pearson Corr./Spearman Corr.)</th>
<th>WNLI (Accuracy)</th>
</tr>
</thead>
<tbody>
<tr>
<td>The median result</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RoBERTa, Liu et al. 2019</td>
<td>90.2/90.2</td>
<td>92.2</td>
<td>92.4/-</td>
<td>–</td>
</tr>
<tr>
<td>RoBERTa, our run</td>
<td>90.5/90.5</td>
<td>91.6</td>
<td>92.0/92.0</td>
<td>56.3</td>
</tr>
<tr>
<td>SSL-Reg (MTP)</td>
<td><strong>90.7/90.7</strong></td>
<td>91.6</td>
<td>92.0/92.0</td>
<td><strong>62.0</strong></td>
</tr>
</tbody>
</table>

The best result

<table>
<thead>
<tr>
<th></th>
<th>MNLI-m/mm (Accuracy)</th>
<th>QQP (Accuracy)</th>
<th>STS-B (Pearson Corr./Spearman Corr.)</th>
<th>WNLI (Accuracy)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RoBERTa, our run</td>
<td>90.7/90.5</td>
<td>91.7</td>
<td>92.3/92.2</td>
<td>60.6</td>
</tr>
<tr>
<td>SSL-Reg (MTP)</td>
<td>90.7/90.5</td>
<td><strong>91.8</strong></td>
<td>92.3/92.2</td>
<td><strong>66.2</strong></td>
</tr>
</tbody>
</table>

Table 13: Ablation study on sentence augmentation types in SSL-Reg (SATP), where SR, RD, RI, and RS denotes synonym replacement, random deletion, random insertion, and random swap respectively. Results are averaged over 5 runs with different random initialization.

<table>
<thead>
<tr>
<th></th>
<th>CoLA</th>
<th>SST-2</th>
<th>RTE</th>
<th>QNLI</th>
<th>MRPC</th>
<th>STS-B</th>
</tr>
</thead>
<tbody>
<tr>
<td>SR+RD+RI+RS</td>
<td>63.6</td>
<td><strong>94.0</strong></td>
<td><strong>74.8</strong></td>
<td>92.2</td>
<td>86.8/90.6</td>
<td><strong>90.6/90.3</strong></td>
</tr>
<tr>
<td>SR+RD+RI</td>
<td>63.4</td>
<td>93.8</td>
<td>72.8</td>
<td>92.1</td>
<td>86.9/90.8</td>
<td>90.6/90.2</td>
</tr>
<tr>
<td>SR+RD</td>
<td>61.6</td>
<td>93.6</td>
<td>72.5</td>
<td><strong>92.2</strong></td>
<td><strong>87.2/91.0</strong></td>
<td><strong>90.6/90.3</strong></td>
</tr>
</tbody>
</table>

run) on 7 out of 9 datasets and achieves the same performance as RoBERTa (our run) on the rest 2 datasets. In terms of best results, SSL-Reg (MTP) performs better than RoBERTa (our run) on 5 out of 9 datasets and achieves the same performance as RoBERTa (our run) on the rest 4 datasets. This further demonstrates the effectiveness of our proposed SSL-Reg approach which uses an MTP-based self-supervised task to regularize the finetuning of RoBERTa.

In SSL-Reg (SATP), we perform an ablation study on different types of sentence augmentation. Results are shown in Table 13, where SR, RD, RI, and RS denote synonym replacement, random deletion, random insertion, and random swap, respectively. SR+RD+RI+RS means that we apply these four types of operations to augment sentences; given an augmented sentence \( a \), we predict which of the four types of operations was applied to an original sentence to create \( a \). SR+RD+RI+RS and SR+RD hold similar meanings. From this table, we make the following observations.

First, as the number of augmentation types increases from 2 (SR+RD) to 3 (SR+RD+RI) then...
to 4 (SR+RD+RI+RS), the performance increases in general. This shows that it is beneficial to have more augmentation types in SATP. The reason is that more types make the SATP task more challenging and solving a more challenging self-supervised learning task can enforce sentence encoders to learn more powerful representations.

Second, SR+RD+RI+RS outperforms SR+RD +RI on 5 out of 6 datasets. This demonstrates that leveraging random swap (RS) for SATP can learn more effective representations of sentences. The reason is: SR, RD, and RI change the collection of tokens in a sentence via synonym replacement, random deletion, and random insertion; RS does not change the collection of tokens, but changes the order of tokens; therefore, RS is complementary to the other three operations; adding RS can bring in additional benefits that are complementary to those of SR, RD, and RI.

Third, SR+RD+RI performs much better than SR+RD on CoLA and is on par with SR+RD on the rest five datasets. This shows that adding RI to SR+RD is beneficial. Unlike synonym replacement (SR) and random deletion (RD) which do not increase the number of tokens in a sentence, RI increases token number. Therefore, RI is complementary to SR and RD and can bring in additional benefits.

5 Conclusions and Future Work

In this paper, we propose to use self-supervised learning to alleviate overfitting in text classification problems. We propose SSL-Reg, which is a regularizer based on SSL and a text encoder is trained to simultaneously minimize classification loss and regularization loss. We demonstrate the effectiveness of our methods on 17 text classification datasets.

For future work, we will use other self-supervised learning tasks to perform regularization, such as contrastive learning, which predicts whether two augmented sentences stem from the same original sentence.

References


Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don’t stop pretraining: Adapt language models to


Xuehai He, Xingyi Yang, Shanghang Zhang, Jinyu Zhao, Yichen Zhang, Eric Xing, and Pengtao Xie. 2020b. Sample-efficient deep learning for covid-19 diagnosis based on CT scans. medRxiv.


David Jurgens, Srijan Kumar, Raine Hoover, Daniel A. McFarland, and Dan Jurafsky. 2018. Measuring the evolution of a scientific field through citation frames. TACL. https://doi.org/10.1162/tacl_a_00028


Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text


Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In EMNLP. https://doi.org/10.18653/v1/D18-1360

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In ACL.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In ACM SIGIR. https://doi.org/10.1145/2766462.2767755


