Fine-tuning Happens in Tiny Subspaces: Exploring Intrinsic Task-specific Subspaces of Pre-trained Language Models

Zhong Zhang[1,2], **Bang Liu**[3,*,†], **Junming Shao**[1,2,†]
[1]University of Electronic Science and Technology of China, Chengdu, China
[2]Shenzhen Institute for Advanced Study, UESTC, Shenzhen, China
[3]Mila & Université de Montréal, Montréal, Canada
zhongzhang@std.uestc.edu.cn,
bang.liu@umontreal.ca,
junmshao@uestc.edu.cn

Abstract

Pre-trained language models (PLMs) are known to be overly parameterized and have significant redundancy, indicating a small degree of freedom of the PLMs. Motivated by the observation, in this paper, we study the problem of re-parameterizing and fine-tuning PLMs from a new perspective: Discovery of intrinsic task-specific subspace. Specifically, by exploiting the dynamics of the fine-tuning process for a given task, the parameter optimization trajectory is learned to uncover its intrinsic task-specific subspace. A key finding is that PLMs can be effectively fine-tuned in the subspace with a small number of free parameters. Beyond, we observe some outlier dimensions emerging during fine-tuning in the subspace. Disabling these dimensions degrades the model performance significantly. This suggests that these dimensions are crucial to induce task-specific knowledge to downstream tasks.

1 Introduction

Pre-trained Language Models (PLMs) have become the de facto methods for various natural language processing (NLP) tasks (Devlin et al., 2019; Radford et al., 2019; Liu et al., 2019). The typical paradigm is to pre-train a big language model on large-scale corpora and then fine-tune the model on small task-specific datasets to adapt to the downstream tasks. Despite the great success of this paradigm, two questions still come to our mind: (1) Why can a PLM with hundreds of millions of parameters be successfully fine-tuned on different downstream tasks using only hundreds or thousands of labeled samples? (2) Do we really need a full fine-tuning of all parameters of a PLM to reach state-of-the-art performance on downstream tasks? In this paper, we try to provide a new viewpoint on the two questions, and claim that: **Fine-tuning happens only in some tiny task-specific subspaces, which can be effectively learned with a small number of free parameters**.

Recent studies have shown that PLMs are highly over-parameterized and robust to pruning (Frankle and Carbin, 2019; Chen et al., 2020; Prasanna et al., 2020; Gordon et al., 2020; Liang et al., 2021, 2022), and can be fine-tuned in parameter-efficient ways (Gong et al., 2022; Zaken et al., 2022; Mahabadi et al., 2021; Li and Liang, 2021). This emerging empirical evidence tends to point to one fact that there exist some intrinsic structures in PLMs that are responsible for inducing task-specific knowledge to downstream tasks. Notably, the recent work (Aghajanyan et al., 2021) provides a promising conclusion that PLMs can be re-parameterized and fine-tuned in random low-dimensional subspaces using random projection, and the dimensionality of the random subspace is orders of magnitude smaller than the dimensionality of the full parameter space. Their findings implicitly suggest the existence of such intrinsic structure in the PLMs, which is, however, understudied. To bridge this gap, we explicitly demonstrate that there exist task-specific low-dimensional subspaces in which PLMs can be effectively fine-tuned.

Inspired by the low dimensional landscape hypothesis (Li et al., 2022a) that a training trajectory of a neural network lies in a low-dimensional subspace, in this work, we thus resort to the fine-tuning trajectory to study the intrinsic task-specific subspaces of PLMs. We show that it is possible to uncover the intrinsic task-specific subspaces with a fine-tuning trajectory by finding its principal directions. The uncovered intrinsic task-specific subspaces usually have very low dimensionalities, but are quite effective in inducing task-specific knowledge. For example, by re-parameterizing the encoder and optimizing only 32 free parameters per-

[*] Canada CIFAR AI Chair.
[†] Corresponding authors.

layer in the intrinsic task-specific subspace, the model allows achieving nearly the same performance as fine-tuning in the full parameter space. Moreover, we further show that the uncovered intrinsic task-specific subspaces have a certain transferability.

Beyond this, we find that the model contains some outlier dimensions with abnormal spikes when fine-tuning in the intrinsic task-specific subspaces instead of a random subspace. Disabling these outlier dimensions degrades the model performance significantly. We believe that this phenomenon is related to the previously discovered outlier dimensions of PLMs (Luo et al., 2021; Kovaleva et al., 2021; Puccetti et al., 2022). However, there are essential differences between them, which we will discuss in the latter section.

By exploring the intrinsic task-specific subspaces of PLMs, the main contributions of this paper are summarized as follows.

1. We interpret the ease of adapting PLMs to downstream tasks as fine-tuning happens in tiny intrinsic task-specific subspaces. Within this interpretation, we propose a method to uncover the subspaces by finding the principal directions of the fine-tuning trajectory.

2. We conduct extensive experiments on the GLUE benchmark using BERT and RoBERTa models to support our claims. We show that the models can be effectively fine-tuned with a very small number of parameters in the uncovered intrinsic task-specific subspaces.

3. We identify some outlier dimensions when fine-tuning in the intrinsic task-specific subspaces, and some empirical analysis is further given.

2 Related Work

Intrinsic Dimensionality. Li et al. (2018) first defined the intrinsic dimension of an objective function in the context of deep learning. They showed that various neural networks can be effectively re-parameterized and trained in random low-dimensional subspaces. Their findings shed light on understanding the high-dimensional landscape of complex neural networks. Following this, Aghajanyan et al. (2021) further measured the intrinsic dimensions of PLMs fine-tuning on downstream tasks. They showed that PLMs have very low intrinsic dimensions ranging from hundreds to thousands. Qin et al. (2021) exploited the idea of intrinsic subspace and proposed a prompt tuning method for efficient training. In addition, the concept of intrinsic dimension is also related to the low-rank approximation of PLMs (Hu et al., 2022; Mahabadi et al., 2021; Chen et al., 2021), but their motivations are entirely different. The former aims to open the black box of models and explore the internal mechanisms of why they are effective, while the latter focuses on developing new methods to train the models efficiently.

Random Projection and Subspace Learning. The random projection has a long history in machine learning research community, and is a key tool to analyze the intrinsic dimension (Li et al., 2018; Aghajanyan et al., 2021). In the context of optimization, Gressmann et al. (2020) proposed a random bases descent algorithm to train neural networks in low-dimensional subspaces. However, the random projection inevitably introduces task-irrelevant information, and is not optimal for subspace learning. We believe that a more compact and task-specific subspace can be found in the model, which is the main motivation of this work. Gur-Ari et al. (2018) empirically found that gradient descent of neural networks happens in a tiny subspace, Li et al. (2022a) further developed a subspace learning algorithm DLDR that dynamically extracts the subspace from the optimization trajectory. Li et al. (2022b) leveraged the DLDR algorithm for adversarial training. However, to the best of our knowledge, there is no research on the discovery of non-random intrinsic task-specific subspace of PLMs.

Outlier Dimensions in Pre-trained Language Models. Multiple studies have identified outlier dimensions in PLMs. Some works were motivated by calibrating the anisotropy behavior of hidden representation of PLMs (Timkey and van Schijndel, 2021; Ding et al., 2022; Luo et al., 2021; Su et al., 2021; Zhang et al., 2020). Another line of work identified certain outlier dimensions in PLMs that are very sensitive to the fine-tuning of downstream tasks (Kovaleva et al., 2021; Puccetti et al., 2022). Disabling these outlier dimensions degrades the model performance significantly. Luo et al. (2021) showed that the outlier dimensions are artefacts derived from positional embeddings and layer normalization. Puccetti et al. (2022) identified a correlation between outlier dimensions and token frequency. It is worth noting that our findings differ largely from previous works in three ways: 1) The outlier dimensions in their context actually refer to output neurons. In our context, an outlier dimension refers to a specific model parameter. In other words, they consider abnormal outputs, while we consider abnormal weights. 2) The ways of identifying outlier dimensions are different. They identify outlier dimensions by examining abnormal outputs, while we find outlier dimensions by examining abnormal updates to weights. 3) The effects of disabling outlier dimensions are different. They show that disabling just one outlier neuron can result in a significant drop in performance. In contrast, disabling the top outlier weight has almost no effect on the model performance. However, the model performance will drop significantly if we disable more outlier weights. The reason for the emergence of these outlier dimensions remains unclear, and we aim to conduct further in-depth analysis in future work.

3 Intrinsic Task-specific Subspaces Discovery in PLMs

3.1 Preliminary: Intrinsic Dimensionality

The intrinsic dimension of an objective landscape is first defined by Li et al. (2018), which is the number of independent optimization variables with regard to minimizing the objective function. However, finding the exact intrinsic dimension is computationally intractable for complex objective functions like deep neural networks. Therefore, a random subspace training method is usually employed to estimate the intrinsic dimension (Li et al., 2018; Aghajanyan et al., 2021).

Formally, let $\boldsymbol{\theta}^D \in \mathbb{R}^D$ be a parameter vector that parameterizes a model $f(\boldsymbol{x};\boldsymbol{\theta})$. Take the BERT-base model as an example, $\boldsymbol{\theta}^D$ represents all BERT's parameters that are flattened into a 110M-dimensional vector. $\boldsymbol{\theta}_0^D \in \mathbb{R}^D$ denotes the initial parameterization, $\boldsymbol{P} \in \mathbb{R}^{D\times d}$ denotes a random projection matrix whose columns form an orthonormal basis for a randomly oriented d-dimensional subspace of $\mathbb{R}^D$, $\boldsymbol{\theta}^d \in \mathbb{R}^d$ denotes a parameter vector in a lower d-dimensional space. The model is fine-tuned in the lower d-dimensional subspace via the following re-parameterization method:

$$\boldsymbol{\theta}^D = \boldsymbol{\theta}_0^D + \boldsymbol{P}\boldsymbol{\theta}^d. \qquad (1)$$

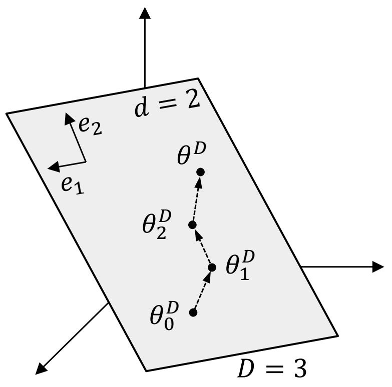

Figure 1: An illustrative example of optimizing a model in the 3-dimensional space, while the optimization trajectory only lies in a 2-dimensional subspace. We call the subspace the intrinsic subspace for the model.

Note that $\boldsymbol{\theta}_0^D$ and $\boldsymbol{P}$ are frozen during the training process, and only $\boldsymbol{\theta}^d$ is trained by the gradient descent. In practice, the re-parameterization can be done in a layer-wise manner to save computational resources (Aghajanyan et al., 2021), and we also follow the layer-wise setting for our analysis.

The intrinsic dimension of a PLM is estimated by grid searching the minimal d that makes the model reach 90% of the full fine-tuning performance. Take the BERT-base model as an example, the intrinsic dimension for fine-tuning on the MRPC dataset is only 1861 (Aghajanyan et al., 2021), which is surprisingly small considering the original model has up to 110 million parameters.

3.2 Finding Intrinsic Task-specific Subspaces

Gur-Ari et al. (2018) showed strong empirical evidence that the gradient dynamically converges to a very small subspace in various large-scale deep-learning scenarios. The subspace is spanned by a few top eigenvectors of the Hessian, and the dimension is equal to the number of data classes. This also indicates that the training trajectory of neural networks lies in a low-dimensional subspace, which is in line with the conclusion of Li et al. (2022a). Considering an illustrative example in Fig. 1, the full parameter space contains three dimensions, but the training trajectory $\{\boldsymbol{\theta}_i^D\}_{i=0,..,t}$ only lies in a 2-dimensional subspace $\mathcal{S}$ spanned by $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$. We call this subspace the intrinsic subspace because it has a minimal degree of freedom (Li et al., 2018) for the objective function to reach the optimum. The aforementioned random subspace can be seen as a naïve estimation of $\mathcal{S}$.

We hypothesize that an intrinsic task-specific subspace exists for each downstream task when fine-tuning a PLM. Generally, it is intractable to search such an intrinsic task-specific subspace directly. However, if our hypothesis is true, the fine-tuning trajectory will lie in a low-dimensional subspace. Thus we can resort to the fine-tuning trajectory to obtain an approximation of the intrinsic task-specific subspace. Specifically, given a fine-tuning trajectory $\{\boldsymbol{\theta}_i^D\}_{i=0,..,t}$ of a PLM on a downstream task, we stack it into a matrix $\boldsymbol{W} \in \mathbb{R}^{t\times D}$, and apply Singular Value Decomposition (SVD) on it.

$$\boldsymbol{W} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T, \tag{2}$$

where $\boldsymbol{\Sigma} \in \mathbb{R}^{t\times t}$ is the singular value matrix, $\boldsymbol{U} \in \mathbb{R}^{t\times t}$ and $\boldsymbol{V} \in \mathbb{R}^{D\times t}$ are two real orthogonal matrices whose columns are left and right singular vectors, respectively[1]. It is worth noting that the columns of $\boldsymbol{V}$ are actually the principal directions of the given trajectory if zero empirical means of columns, and these directions constitute an orthonormal basis of the subspace in which the trajectory lies. Theoretically, a $(t-1)$-dimensional subspace needs only t independent points to determine. We can regard this subspace as an approximation of the intrinsic task-specific subspace whose dimension is equal to the number of points in the trajectory. Thus, we can replace the random projection matrix $\boldsymbol{P}$ in Eq. (1) with $\boldsymbol{V}$ to re-parameterize the model.

3.3 Fine-tuning in Intrinsic Task-specific Subspaces

Given an approximated intrinsic task-specific subspace $\boldsymbol{V}$, we reformulate Eq. (1) by letting the model train in the subspace as follows.

$$\boldsymbol{\theta}^D = \boldsymbol{\theta}_0^D + \boldsymbol{V}\boldsymbol{\theta}^t. \tag{3}$$

In our early exploration, we can achieve good performance close to full fine-tuning by Eq. (3). However, the performance is not stable, and sensitive to the initialization of $\boldsymbol{\theta}^t$. To solve this problem, we propose an ensemble-like method that combines multiple $\boldsymbol{\theta}^t$ of different initialization to reduce variance, which is as follows.

$$\boldsymbol{\theta}^D = \boldsymbol{\theta}_0^D + \boldsymbol{V}\sum_{i=1}^{h}\frac{1}{h}\boldsymbol{\theta}^{t(i)}, \tag{4}$$

[1]We assume $t \ll D$ and thus compact SVD is applied.

where h is the number of vectors to combine, and we set it as 16 in this paper. Note that although the ensemble increases the number of parameters to optimize, it does not change the instrinsic dimensionality of the subspace (i.e., the degree of freedom).

In the following experimental evaluation, we will investigate subspace fine-tuning in both transductive and inductive settings to verify our hypotheses. The former is to verify the existence of intrinsic task-specific subspaces when fine-tuning PLMs on the downstream tasks, and the effectiveness of our method to uncover the subspaces. The latter further examines how well the intrinsic task-specific subspaces can be transferred to other similar tasks.

4 Experiment and Analysis

4.1 Experimental Settings

Datasets and models. We evaluate the performance of the methods on the commonly used GLUE benchmark (Wang et al., 2018; Warstadt et al., 2019; Socher et al., 2013; Dolan and Brockett, 2005; Cer et al., 2017; Williams et al., 2018; Rajpurkar et al., 2016). For evaluation metrics, we report the matched accuracy for MNLI, Matthews correlation for CoLA, Pearson correlation for STS-B, and accuracy for other tasks. We choose the publicly available pre-trained language models RoBERTa-base (Liu et al., 2019) and BERT-base-cased (Devlin et al., 2019) for analysis. All experimental results are averaged over 5 runs of different seeds.

Implementation details. Our implementation is based on HuggingFace's Transformers toolkit (Wolf et al., 2020). We first need to produce a set of fine-tuning trajectories of GLUE tasks for calculating projection matrices. We use the default script in the toolkit for fine-tuning, and save a checkpoint every epoch to obtain optimization trajectories. We set the trajectory length to 32 except for the MNLI dataset, which is set to 64 since it is the largest dataset and needs more parameters to fit. We flatten all parameters in an encoder layer into a wide vector, and then stack all vectors of different checkpoints into a matrix to perform SVD. We compute independent projection matrices for all layers, resulting in 12 projection matrices. For transductive subspace fine-tuning, the projection matrix is calculated from the same task,

	CoLA	MRPC	SST-2	STS-B	QQP	MNLI	QNLI	RTE	Avg.
BERT-Full	59.37	**84.46**	**91.95**	89.08	**91.07**	**83.39**	**90.77**	66.93	**82.13**
BERT-Freeze	27.52	69.66	88.81	78.35	84.48	71.55	81.61	56.46	69.81
BERT-Random	37.89	70.78	89.47	81.41	85.86	72.91	83.38	58.63	72.54
BERT-Intrinsic	**60.27**	84.31	89.93	**89.51**	89.73	81.21	87.73	**67.00**	81.21
RoBERTa-Full	61.04	**89.31**	**94.29**	**90.70**	**91.72**	**87.23**	**92.48**	76.68	**85.43**
RoBERTa-Freeze	0.00	68.38	85.32	15.69	82.81	71.16	79.11	53.86	57.04
RoBERTa-Random	27.58	68.38	91.45	75.47	86.33	77.10	84.49	58.27	71.13
RoBERTa-Intrinsic	**61.07**	87.21	92.43	89.43	90.18	85.53	90.57	**78.77**	84.40

Table 1: Transductive intrinsic subspace fine-tuning on the GLUE benchmark. *Full* denotes fine-tuning in the full-parameter space. *Freeze* denotes fine-tuning with the encoder frozen. *Random* denotes fine-tuning in a random subspace. *Intrinsic* denotes fine-tuning in the intrinsic task-specific subspaces. The subspace dimension is set to 32 except MNLI is 64. The best results are marked in bold, and the second-best results are underlined.

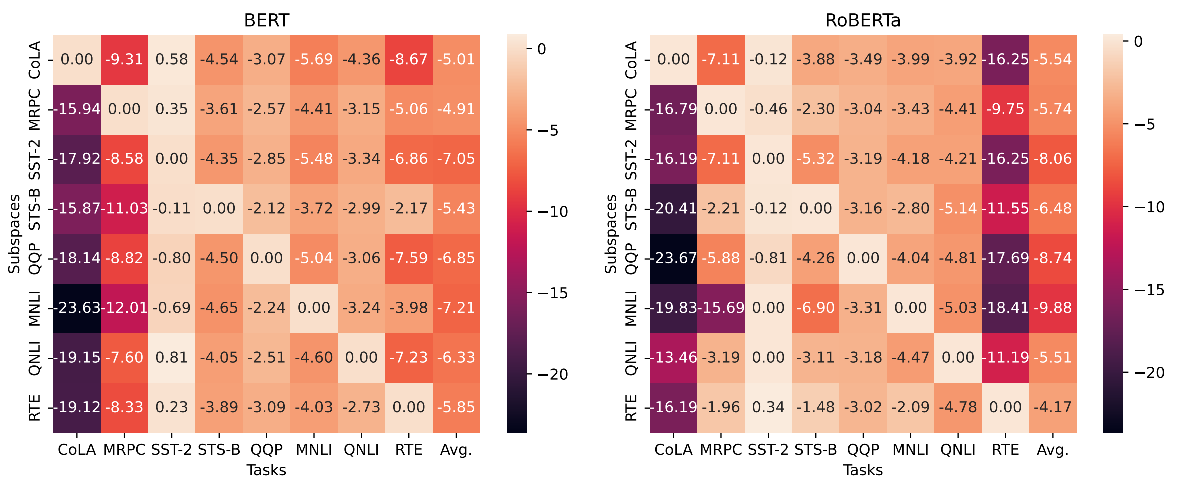

Figure 2: Inductive intrinsic subspace fine-tuning on the GLUE benchmark. Columns are the target tasks to be fine-tuned, and rows are source tasks that provide the transferred subspaces. Numbers in cells are performance drop of fine-tuning target tasks with the subspaces provided by source tasks. The last column is the average of other columns. Note that the numbers cannot be compared across columns because they are in different metrics.

while for inductive subspace fine-tuning, it is calculated from other tasks. We only re-parameterize the encoder layers into the subspaces and leave the embedding layer and the last classification layer in their original parameter space. We freeze the initial model $\boldsymbol{\theta}_0^D$ and the projection matrix $\boldsymbol{V}$, and only tune the low-dimensional vector $\boldsymbol{\theta}^t$. We keep the learning rate of the embedding and classification layers unchanged and set the learning rate of $\boldsymbol{\theta}^t$ to 0.01.

4.2 Transductive Intrinsic Subspace Fine-tuning

Table 1 summarizes the experimental results. We can see that freezing the encoder significantly degrades the model performance as it serves as a naïve baseline (Note that it implies fine-tuning in the null space, i.e., $\boldsymbol{V}\boldsymbol{\theta}^t = \mathbf{0}$, which brings no information to update the model). For intrinsic subspace fine-tuning, we can clearly see that it shows comparable performance to the full fine-tuning across all GLUE tasks and models. In contrast, random projection only yields a marginal improvement over the baseline, and significantly underperforms intrinsic subspace fine-tuning.

From these empirical results, we first conclude that PLMs can be re-parameterized and fine-tuned in some low-dimensional subspaces. Secondly, there exist some subspaces in which the PLMs can most effectively adapt to downstream tasks, and we can uncover these subspaces by finding the principal directions of fine-tuning trajectories in the full parameter space. This conclusion in turn suggests that fine-tuning of PLMs happens in tiny

	CoLA	MRPC	SST-2	STS-B	QQP	MNLI	QNLI	RTE	Avg.
BERT-Full	59.37	**84.46**	**91.95**	89.08	91.07	83.39	90.77	66.93	82.13
BERT-Random	32.49	70.15	88.65	79.29	84.84	71.75	82.29	57.11	70.82
BERT-Zeroshot	35.35	78.09	91.06	85.17	87.57	75.29	84.01	**75.23**	76.47
BERT-Unified	**61.58**	84.41	91.06	**89.71**	**91.27**	**83.85**	**90.97**	67.00	**82.48**
RoBERTa-Full	61.04	**89.31**	**94.29**	90.70	91.72	**87.23**	**92.48**	76.68	85.43
RoBERTa-Random	0.00	68.38	89.47	27.60	84.51	73.16	82.10	54.44	59.96
RoBERTa-Zeroshot	32.93	80.44	90.60	83.10	87.12	78.76	84.46	67.12	75.57
RoBERTa-Unified	**63.80**	89.12	93.55	**90.88**	**91.85**	87.20	92.36	**77.91**	**85.83**

Table 2: Intrinsic subspace fine-tuning in the unified task subspace. *Random* denotes fine-tuning in a random subspace (dim=8). *Zeroshot* denotes fine-tuning in the unified task subspace with itself removed (dim=7). *Unified* denotes fine-tuning in the unified task subspace (dim=8).

subspaces, which provides an explanation of the ease of adapting PLMs to downstream tasks.

4.3 Inductive Intrinsic Subspace Fine-tuning

Next, we conduct inductive intrinsic subspace fine-tuning to examine the transferability of the discovered subspaces. We generally follow the same training protocol as in the last section, except that we replace the projection matrices with the ones calculated from other tasks.

We can observe the performance drop using transferred task subspaces in Fig. 2. Generally, we can see that even though the models are fine-tuned in transferred subspaces, they still outperform the random subspace baseline, which suggests the transferability of intrinsic task-specific subspaces.

The transferability of subspaces seems to correlate with the scale of the transferred task. For example, big datasets like SST-2, QQP, MNLI and QNLI underperform small datasets like CoLA, MRPC, STS-B, and RTE in providing subspaces. This is because the intrinsic task-specific subspaces of complex tasks have higher dimensions and need more parameters to estimate.

When comparing within one column, we can see significant difference between distinct subspaces used for fine-tuning one task. We assume similar tasks may have substantial subspace intersections and thus be easier to transfer. Still, this claim needs further analysis to confirm, we will leave it further study since transferability is not the main focus of this paper. In summary, we empirically show that the intrinsic task-specific subspace has a certain transferability.

4.4 Unified Intrinsic Task Subspace

Qin et al. (2021) showed that a unified low-dimensional intrinsic task subspace can be constructed by a multi-task prompt tuning method. In our case, we can also construct a unified subspace by stacking the fine-tuning trajectories of different tasks into a matrix, and applying SVD on it. Specifically, we sample one checkpoint for each task and gather them to calculate the unified subspace, which forms an 8-dimensional subspace. And we additionally calculate a zero-shot subspace of a task for comparison, which is calculated by excluding the checkpoint of this task. The results are given in Table 2. We can see that the models can be effectively fine-tuned in the unified subspace. For the zero-shot setting, the model performance decreases significantly, but still outperforms the random baseline.

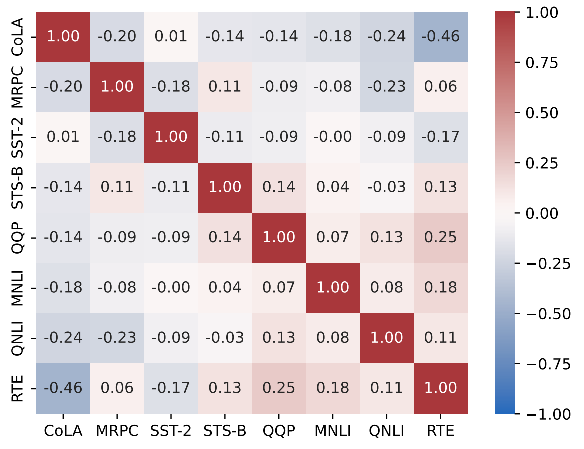

Figure 3: The cosine similarities between the low-dimensional parameter vectors $\boldsymbol{\theta}^t$ of different tasks in the unified intrinsic task subspace. Similarities are averaged over layers and ensembles.

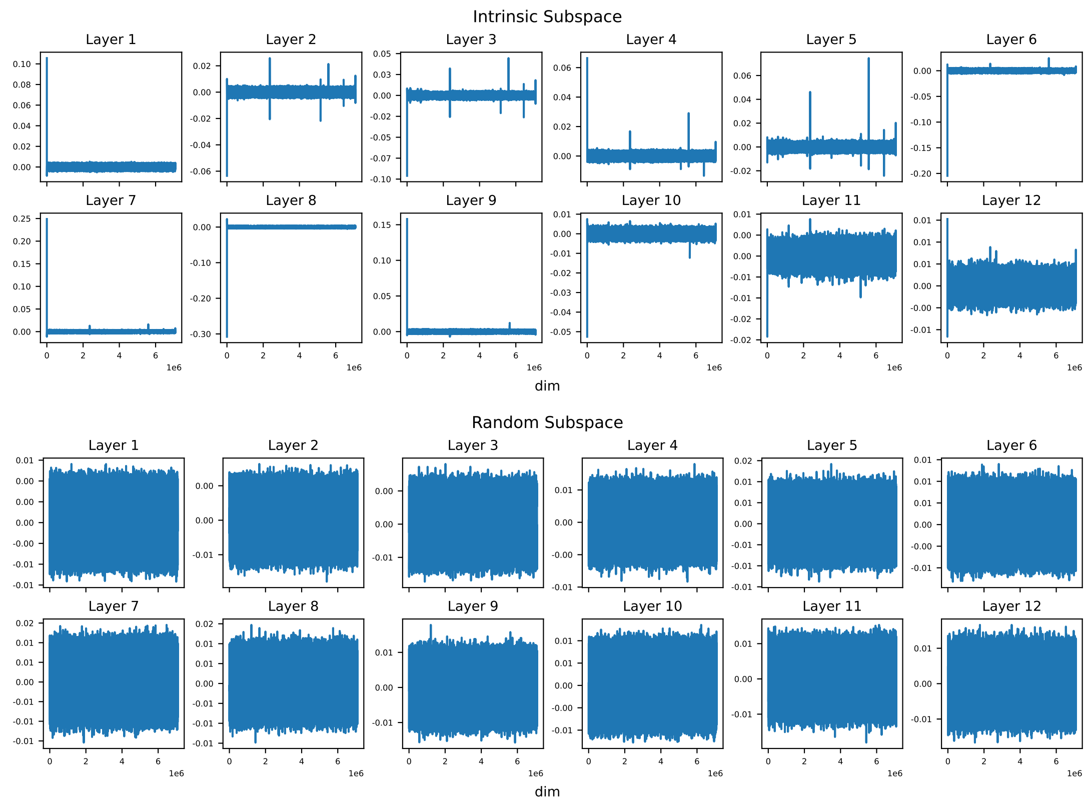

Figure 4: Visualization of product of $\boldsymbol{V}\boldsymbol{\theta}^t$ using the BERT model to fine-tune in the intrinsic task-specific subspace (top) and a random subspace (bottom) on the MRPC dataset. Outlier dimensions appear in the intrinsic subspace but not in a random subspace.

Next, we take the BERT model as an example and examine the low-dimensional parameter vector $\boldsymbol{\theta}^t$ learned within the unified intrinsic subspace. We calculate the cosine similarities between the $\boldsymbol{\theta}^t$ vectors corresponding to different tasks and present the results in Fig. 3. As shown in the figure, the cosine similarities between different tasks are significantly low, indicating that the unified intrinsic subspace contains disentangled knowledge distributed in different dimensions, and the low-dimensional parameter vector $\boldsymbol{\theta}^t$ serves as an (unnormalized) probability distribution to induce task-specific knowledge.

Based on these empirical findings, we conclude that a unified intrinsic task subspace is feasible and it contains disentangled knowledge. However, in-domain knowledge still plays a crucial role in forming the subspace as we can see that the zero-shot setting still has a large perform gap.

4.5 Outlier Dimensions

We find that PLMs have a small number of outlier dimensions exhibiting abnormal spikes when fine-tuning in the intrinsic task-specific subspaces. We examine each dimension of the product of $\boldsymbol{V}\boldsymbol{\theta}^t$ and consider the dimension whose absolute value is greater than a threshold as outlier. Note that the product of $\boldsymbol{V}\boldsymbol{\theta}^t$ is the learned parameter update in the full parameter space and we re-parameterize the encoder of the PLM layer-wisely, thus it is a vector with the dimension equal to the number of all parameters of an encoder layer.

It is important to note that the outlier dimension in our context is different from the previous studies (Kovaleva et al., 2021; Luo et al., 2021; Puccetti et al., 2022). Previous studies use the outlier dimension to refer to the output channel (768 dimensions for BERT-base). In our context, we flatten all parameters of a layer into a vector (7,087,872 dimensions for BERT-base). Then an outlier dimension refers to a specific parameter weight in the layer. We use the BERT model and MRPC dataset for illustration, and visualize the product of $\boldsymbol{V}\boldsymbol{\theta}^t$ in Fig. 4 to show the outlier patterns. As we can see from the figure, when fine-tuning in the intrinsic task-specific subspace, the outlier patterns exist in all layers. In contrast, these outlier patterns disappear when fine-tuning in a random subspace.

	CoLA	MRPC	SST-2	STS-B	QQP	MNLI	QNLI	RTE
BERT-Full	59.37	84.46	91.95	89.08	91.07	83.39	90.77	66.93
BERT-Random	57.27	84.46	91.79	88.66	90.66	83.68	90.41	64.48
BERT-Outlier	**0.00**	**68.38**	**50.92**	**0.00**	**63.18**	**33.64**	**49.89**	**52.71**
RoBERTa-Full	61.04	89.31	94.29	90.70	91.72	87.23	92.48	76.68
RoBERTa-Random	58.80	87.65	93.95	89.52	91.29	87.76	92.61	68.88
RoBERTa-Outlier	**0.00**	**70.49**	**50.92**	**28.05**	**63.67**	**36.15**	**49.89**	**52.71**

Table 3: Evaluation on the GLUE benchmark when the outlier dimensions are zeroed. The results with the most performance loss are marked in bold.

Model component	**Layer**	**# of outliers each layer**
attention.self.query.weight	1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12	3, 1, 1, 1, 4, 4, 8, 3, 3, 2, 4
attention.self.query.bias	1	1
attention.self.key.bias	10, 11	2, 1
attention.output.LayerNorm.weight	1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12	1, 2, 3, 5, 4, 1, 2, 4, 1, 3, 2
attention.output.LayerNorm.bias	1, 2, 3	1, 1, 1
intermediate.dense.weight	1, 12	2, 1
output.dense.weight	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11	2, 6, 5, 4, 2, 4, 3, 2, 3, 4, 4
output.LayerNorm.weight	5, 6, 7, 12	4, 1, 1, 3

Table 4: Sampled outlier dimensions in the BERT model. The left column shows the model component containing outlier dimensions. The middle column shows the layer where the model component contains outlier dimensions. The right column shows the number of outlier dimensions in the corresponding layer.

This phenomenon is universal for different models and different datasets.

To investigate the effect of the outlier dimensions on the models, we disable them by setting them to zero and examine how this affects model performance. We first disable the top outlier dimension of each encoder layer and fine-tune the model in the full parameter space, which has almost no impact on model performance. This result is not surprising because disabling only one weight in a layer definitely has a negligible effect on the output than disabling an output channel as the previous studies do. We continue to disable more outlier dimensions, and these deviating at least 3σ from the mean are disabled. Approximately 0.3% of encoder parameters are disabled. We also randomly sample and disable the same number of dimensions for comparison, and the results are shown in Table 3. We can see that disabling outlier dimensions degrades the model performance significantly while disabling random dimensions does not.

Next, we qualitatively examine the positions in which the outlier dimensions emerge. We sample each layer's top 10 outlier dimensions and record their positions in Table 4. We can see that the outlier dimensions are ubiquitous in various model components. Then, we identify one outlier dimension $O1$ that consistently produces high-magnitude weights in almost all BERT layers. Furthermore, we find that there is a considerable overlap in the outlier dimensions of each layer, which suggests that these dimensions can propagate through layers.

Why do outlier dimensions emerge? Previous studies came up with several explanations like high-magnitude scaling factors (Kovaleva et al., 2021), LayerNorm and residual connection (Luo et al., 2021), and unbalanced token frequency (Puccetti et al., 2022). However, these explanations cannot apply to our case because the definitions of the outlier dimension are different. Recall that our approach to identifying outlier dimensions is actually examining re-parameterized parameter updates given the intrinsic task-specific subspace. The magnitude of the updates represents the importance of corresponding parameters with respect to solving the task. We have reason to believe that these dimensions play an important role in constituting the intrinsic subspace and are crucial to induce task-specific knowledge to adapt to downstream tasks.

5 Conclusion

In this paper, we claim that the fine-tuning of PLMs happens in tiny subspaces. To uncover such intrinsic task-specific subspaces, we exploit the fine-tuning trajectory to find its main direction. Our empirical experiments show that PLMs can effectively adapt to downstream tasks when re-parameterizing and training in the found subspaces, which well explains the ease of adapting PLMs to downstream tasks. Furthermore, we find outlier dimensions in PLMs during the subspace training. We consider that these dimensions are crucial to induce task-specific knowledge to downstream tasks. Still, we need further in-depth analysis to understand the reasons and impact of the emergence of outlier patterns.

Limitations

Despite the insights obtained through our analysis, certain limitations persist, which we outline in this section.

With respect to the re-parameterization of parameters as presented in Eq. (3), we adopted the layer-wise setting as proposed by Aghajanyan et al. (2021) in order to alleviate memory and computational burdens. Nonetheless, such a setting restricts us to only identifying local subspaces, rather than discovering global subspaces within the entire parameter space of a pre-trained language model. The existence of a task-specific global subspace is yet to be ascertained. If such a subspace does exist, the correlation between this global subspace and the identified local subspaces needs to be explored in future research.

In terms of experimental settings, the evaluation tasks are limited to natural language understanding tasks, with a lack of natural language generation tasks. On model architecture, decoder-only (e.g., GPT) and encoder-decoder (e.g., T5) models are not included. On model scale, we use basic-size models rather than large ones due to limited computational resources. Consequently, the conclusions drawn in this study may not be applicable to the above situations.

The analysis presented in Section 4.5 demonstrates that pre-trained language models exhibit a small number of outlier dimensions when fine-tuning in the intrinsic task-specific subspaces. Although we have observed a significant decline in model performance when disabling these dimensions, the underlying mechanism responsible for the emergence of these outlier dimensions remains unclear.

Acknowlegments

This work is supported by the Sichuan key research program (22ZDYF3388), Fundamental Research Funds for the Central Universities (ZYGX2019Z014), National Natural Science Foundation of China (61976044, 52079026), Fok YingTong Education Foundation for Young Teachers in the Higher Education Institutions of China (161062), the Canada CIFAR AI Chair Program, and the Canada NSERC Discovery Grant (RGPIN-2021-03115).

References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 7319–7328.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation*, pages 1–14.

Patrick H. Chen, Hsiang-Fu Yu, Inderjit S. Dhillon, and Cho-Jui Hsieh. 2021. DRONE: data-aware low-rank compression for large NLP models. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 29321–29334.

Tianlong Chen, Jonathan Frankle, Shiyu Chang, Sijia Liu, Yang Zhang, Zhangyang Wang, and Michael Carbin. 2020. The lottery ticket hypothesis for pre-trained BERT networks. In *Advances in Neural Information Processing Systems 33*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Yue Ding, Karolis Martinkus, Damian Pascual, Simon Clematide, and Roger Wattenhofer. 2022. On isotropy calibration of transformer models. In *Proceedings of the Third Workshop on Insights from Negative Results in NLP*, pages 1–9.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing*.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *The 7th International Conference on Learning Representations*.

Zhuocheng Gong, Di He, Yelong Shen, Tie-Yan Liu, Weizhu Chen, Dongyan Zhao, Ji-Rong Wen, and Rui Yan. 2022. Finding the dominant winning ticket in pre-trained language models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1459–1472.

Mitchell Gordon, Kevin Duh, and Nicholas Andrews. 2020. Compressing BERT: Studying the effects of weight pruning on transfer learning. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 143–155.

Frithjof Gressmann, Zach Eaton-Rosen, and Carlo Luschi. 2020. Improving neural network training in low dimensional random bases. In *Advances in Neural Information Processing Systems 33*.

Guy Gur-Ari, Daniel A Roberts, and Ethan Dyer. 2018. Gradient descent happens in a tiny subspace. *arXiv preprint arXiv:1812.04754*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations*.

Olga Kovaleva, Saurabh Kulshreshtha, Anna Rogers, and Anna Rumshisky. 2021. BERT busters: Outlier dimensions that disrupt transformers. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021*, pages 3392–3405.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. 2018. Measuring the intrinsic dimension of objective landscapes. In *International Conference on Learning Representations*.

Tao Li, Lei Tan, Zhehao Huang, Qinghua Tao, Yipeng Liu, and Xiaolin Huang. 2022a. Low dimensional trajectory hypothesis is true: Dnns can be trained in tiny subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Tao Li, Yingwen Wu, Sizhe Chen, Kun Fang, and Xiaolin Huang. 2022b. Subspace adversarial training. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13399–13408.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 4582–4597.

Chen Liang, Haoming Jiang, Simiao Zuo, Pengcheng He, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Tuo Zhao. 2022. No parameters left behind: Sensitivity guided adaptive learning rate for training large transformer models. In *The Tenth International Conference on Learning Representations*.

Chen Liang, Simiao Zuo, Minshuo Chen, Haoming Jiang, Xiaodong Liu, Pengcheng He, Tuo Zhao, and Weizhu Chen. 2021. Super tickets in pre-trained language models: From model compression to improving generalization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 6524–6538.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. In *arXiv preprint arXiv:1907.11692*.

Ziyang Luo, Artur Kulmizev, and Xiaoxi Mao. 2021. Positional artefacts propagate through masked language model embeddings. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 5312–5327.

Rabeeh Karimi Mahabadi, James Henderson, and Sebastian Ruder. 2021. Compacter: Efficient low-rank hypercomplex adapter layers. In *Advances in Neural Information Processing Systems 34*, pages 1022–1035.

Sai Prasanna, Anna Rogers, and Anna Rumshisky. 2020. When BERT plays the lottery, all tickets are winning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 3208–3229.

Giovanni Puccetti, Anna Rogers, Aleksandr Drozd, and Felice Dell'Orletta. 2022. Outlier dimensions that disrupt transformers are driven by frequency. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1286–1304.

Yujia Qin, Xiaozhi Wang, Yusheng Su, Yankai Lin, Ning Ding, Zhiyuan Liu, Juanzi Li, Lei Hou, Peng Li, Maosong Sun, et al. 2021. Exploring low-dimensional intrinsic task subspace via prompt tuning. *arXiv preprint arXiv:2110.07867*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642.

Jianlin Su, Jiarun Cao, Weijie Liu, and Yangyiwen Ou. 2021. Whitening sentence representations for better semantics and faster retrieval. *arXiv preprint arXiv:2103.15316*.

William Timkey and Marten van Schijndel. 2021. All bark and no bite: Rogue dimensions in transformer language models obscure representational quality. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4527–4546.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1112–1122.

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

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. 2022. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 1–9.

Zhong Zhang, Chongming Gao, Cong Xu, Rui Miao, Qinli Yang, and Junming Shao. 2020. Revisiting representation degeneration problem in language modeling. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 518–527.

A Appendix

A.1 Hyperparameters

We first fine-tune the BERT and RoBERTa models for calculating projection matrices. We use the fine-tuning script in the Transformers toolkit[2]. All hyperparameters remain default except for the number of epochs, which is set to 32 and 64 for the MNLI and all other tasks, respectively. For intrinsic subspace fine-tuning, the dimensionality of $\boldsymbol{\theta}^t$ is set to 32 and 64 for the MNLI and all other tasks, respectively. The learning rate of $\boldsymbol{\theta}^t$ is set to 0.01. The number of ensembles h is set to 16. Other hyperparameter are the same as those in the script. All experimental results are averaged over 5 runs of different seeds. Each experiment is conducted on a single GeForce RTX 2080Ti GPU with environment of Pytorch 1.11.0 + CUDA 11.3.1.

A.2 Ablation study

We conduct an ablation experiment over the number of dimensions of the subspaces. The results are given in Table 5 and Table 6. The performance increases as the number of dimensions increases.

Tasks	dim=8	dim=16	dim=32
CoLA	54.06	57.17	**60.27**
MRPC	75.05	77.94	**84.31**
SST-2	89.52	**90.05**	89.93
STS-B	87.95	89.02	**89.51**
QQP	87.61	89.12	**89.73**
MNLI	76.93	78.48	**78.70**
QNLI	86.54	86.83	**87.73**
RTE	65.41	66.07	**67.00**

Table 5: Ablation study for the BERT model.

Tasks	dim=8	dim=16	dim=32
CoLA	58.04	60.27	**61.07**
MRPC	75.59	78.20	**87.21**
SST-2	91.93	92.34	**92.43**
STS-B	84.10	88.10	**89.43**
QQP	87.58	89.25	**90.18**
MNLI	79.96	81.77	**82.32**
QNLI	89.35	89.14	**90.57**
RTE	74.30	78.56	**78.77**

Table 6: Ablation study for the RoBERTa model.

[2] https://github.com/huggingface/transformers/tree/main/examples/pytorch/text-classification

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section Limitations.

☑ A2. Did you discuss any potential risks of your work?
Section Limitations.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract, Section 1.

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 4.

☑ B1. Did you cite the creators of artifacts you used?
Section 4.

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
We use open-source artifacts which can be used for academic research purposes.

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
We use the artifacts in compliance with their licenses.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
We use the open-source GLUE dataset which does not contain sensitive information.

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4.

C ☑ Did you run computational experiments?

Section 4.

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☐ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
No response.

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.