A Closer Look at How Fine-tuning Changes BERT

Yichu Zhou
University of Utah

Vivek Srikumar
University of Utah

Abstract

Given the prevalence of pre-trained contextualized representations in today's NLP, there have been several efforts to understand what information such representations contain. A common strategy to use such representations is to fine-tune them for an end task. However, how fine-tuning for a task changes the underlying space is less studied. In this work, we study the English BERT family and use two probing techniques to analyze how fine-tuning changes the space. Our experiments reveal that fine-tuning improves performance because it pushes points associated with a label away from other labels. By comparing the representations before and after fine-tuning, we also discover that fine-tuning does not change the representations arbitrarily; instead, it adjusts the representations to downstream tasks while preserving the original structure. Finally, using carefully constructed experiments, we show that fine-tuning can encode training sets in a representation, suggesting an overfitting problem of a new kind.

1 Introduction

Transformer-based masked language models (e.g., Devlin et al., 2019; Lan et al., 2020) form the basis of state-of-the-art results across NLP. The relative opacity of these models has prompted the development of many probes that investigate linguistic regularities captured in them (e.g., Kovaleva et al., 2019; Conneau et al., 2018; Jawahar et al., 2019).

Broadly speaking, there are two ways to use a pre-trained representation Peters et al. (2019): as a fixed feature extractor (where the pre-trained weights are frozen), or by fine-tuning it for a task. The probing literature has largely focused on the former (e.g., Kassner and Schütze, 2020; Perone et al., 2018; Yaghoobzadeh et al., 2019; Krasnowska-Kieraś and Wróblewska, 2019; Wallace et al., 2019; Pruksachatkun et al., 2020). Some previous work (Merchant et al., 2020; Mosbach et al., 2020b; Hao et al., 2020) does provide insights about fine-tuning: fine-tuning changes higher layers more than lower ones and linguistic information is not lost during fine-tuning. However, relatively less is understood about how fine-tuning affects the underlying representation, and why fine-tuning invariably seems to improve task performance.

In this work, we investigate the process of fine-tuning of representations using the English BERT family (Devlin et al., 2019). Specifically, we ask:

1. Why does fine-tuning improve performance?
2. How does fine-tuning change the underlying geometric structure that the representation uses to encode lexical items?

We apply two probing techniques, classifier-based probing (Kim et al., 2019; Tenney et al., 2019) and DIRECTPROBE (Zhou and Srikumar, 2021), on variants of BERT representations that are fine-tuned on four NLP tasks: part-of-speech tagging, dependency head prediction, and preposition supersense role & function prediction.

Our analysis confirms previous findings about fine-tuning. In addition, it also reveals the following new findings:

- *Fine-tuning simplifies the underlying representation by grouping the points with the same label into one cluster.* It is easier to linearly separate labels with fine-tuned representations than the untuned ones.

- *Fine-tuning improves task performance because it pushes the clusters of points representing different labels away from each other, introducing a large separating regions between labels.* Rather than simply scaling the points, clusters move in diferent directions and with different extents (measured by Euclidean distance). But overall, these clusters become distant compared with the untuned representation. The enlarged region between groups admits a bigger set of classifiers that can separate them, leading to better generalization.
- *Fine-tuning does not change the representation arbitrarily.* Indeed, it largely preserves the relative positions of the clusters, while reconfiguring the space for downstream tasks.
- *Fine-tuning can memorize a training set, and thus admits the risk of overfitting.* Fine-tuning almost always improves task performance. A deeper analysis shows that fine-tuning memorizes the training set for the task and introduces a divergence between the training set and unseen examples.
- *Lower layers do change during fine-tuning, but to a much lesser extent than the higher ones.* Points from lower layers move in a smaller region compared to the higher layers.

These findings help us understand what is happening during fine-tuning and justify why fine-tuned representations can lead to such improvements on many NLP tasks.

2 Preliminaries: Probing Methods

In this work, we will probe representations in the BERT family at various points during and after fine-tuning. As a first step, let us look at the two supervised probes we will employ: a classifier-based probe (Tenney et al., 2019) to assess how well a representation supports classifiers for different tasks, and DIRECTPROBE (Zhou and Srikumar, 2021) to analyze the geometry of the representation.

2.1 Classifiers as Probes

Trained classifiers are perhaps the most commonly used probes in the literature (e.g. Liu et al., 2019; Hewitt and Manning, 2019; Kim et al., 2019). To understand how well a representation encodes the labels for a task, a probing classifier is trained over the representations.

For all our experiments, we use two-layer neural networks as our probe classifiers. We use cross-validation to choose the best hyperparameters. Each best classifier is trained five times with different initializations. We report the average accuracy and standard deviation for each classifier.

The two-layer neural network has hidden layer size of $(32, 64, 128, 256) \times (32, 64, 128, 256)$. The weight of regularizer ranges from 10^{-7} to 10^{0}. All neural networks use ReLUs as the activation function for the hidden layer and are optimized by Adam (Kingma and Ba, 2015). We set the maximum iterations to be 1000.

Classifier probes aim to measure how well a contextualized representation captures a linguistic property. The classification performance can give us a direct assessment of the effect of fine-tuning.

2.2 DirectProbe: Probing the Geometric Structure

Classifier probes treat the representation as a black box and only focus on the final task performance, which does not reveal what changes in the underlying representation during the fine-tuning. Our second probing strategy uses DIRECTPROBE[1], a recently proposed technique which analyzes embeddings from a geometric perspective by clustering.

DIRECTPROBE seeks to characterize not one, but *all* decision boundaries in a representation that are consistent with a training set for a task. It approximates this set of decision boundaries by a supervised

[1]We use the DIRECTPROBE implementation from `https://github.com/utahnlp/DirectProbe`. We use DIRECTPROBE with default settings.

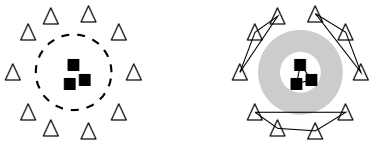


Figure 1: Using the clustering to approximate the set of all decision boundaries. The left subfigure is a simple binary classification problem with a dashed circular decision boundary. The right subfigure is the result of DIRECTPROBE where the gray area is the region that a separator must cross. The connected points represent the clusters that DIRECTPROBE produces.

clustering algorithm. For a given labeling task, DIRECTPROBE returns a set of clusters such that each cluster only contains the points with the same label, and there are no overlaps between the convex hulls of these clusters. Any decision boundary must cross the regions between the clusters with different labels (showed in Figure 1).

Different from the representational similarity analysis (Laakso and Cottrell, 2000; Abnar et al., 2019; Chrupała and Alishahi, 2019; Gauthier and Levy, 2019), DIRECTPROBE requires a representation and a given labeling task. Since fine-tuning a contextualized representation creates different representations for different tasks, it is reasonable to probe the representation based on a given task.

The clusters provided by DIRECTPROBE allow us to measure three properties of interest.

1. Number of Clusters: The number of clusters indicates the linearity of the representation for a task. If the number of clusters equals the number of labels, then examples with the same label are grouped into one cluster; a simple linear multi-class classifier will suffice. In contrast, if the number of clusters is more than the number of labels, then at least two clusters of examples with the same label can not be grouped together (as in Figure 1, right). A non-linear classifier is required to separate labels from each other in this scenario.

2. Distances between Clusters: Distances[2] between clusters can reveal the internal structure of a representation. By tracking these distances, we can show how the representation changes during the fine-tuning. To compute these distances, we use the fact that each cluster represents a convex object. This allows us to use max-margin separators to compute distances. We train a linear SVM (Chang and Lin, 2011) to find the maximum margin separator and compute its margin. The distance between the two clusters is twice the margin.

3. Spatial Similarity: Distances between clusters can also be used to compare the spatial similarity of two representations. Intuitively, if two representations have similar relative distances between clusters, the representations themselves are similar to each other.

We use these distances to construct a *distance vector* $\mathbf{v}$ for a representation, where each element $\mathbf{v}_i$ is the distance between the clusters of a pair of labels. With n labels in a task, the size of $\mathbf{v}$ is $\frac{n(n-1)}{2}$. This construction works only when the number of clusters equals the number of labels (i.e., the dataset is linearly separable under the representation). We found that the construction is valid for most representations. As a measure of the similarity of two representations for a labeling task, we compute the Pearson correlation coefficient between their distance vectors. Note that this coefficient can also be used to measure the similarity between two labeled datasets with respect to the same representation. We exploit this observation to analyze the divergence between training and test sets for fine-tuned representations (§4.2).

3 Experimental Setup

In this section, we describe the representations and tasks we will encounter in our experiments.

[2]We use Euclidean distance throughout this work.

	Layers	#heads	Dim	#Param
$\text{BERT}_{\text{tiny}}$	2	2	128	4.4M
$\text{BERT}_{\text{mini}}$	4	4	256	11.3M
$\text{BERT}_{\text{small}}$	4	8	512	29.1M
$\text{BERT}_{\text{medium}}$	8	8	512	41.7M
$\text{BERT}_{\text{base}}$	12	12	768	110.1M

Table 1: Statistics of five different BERT models.

3.1 Representations

We investigate several models from the BERT family (Devlin et al., 2019; Turc et al., 2019). These models all share the same basic architecture but with different capacities, i.e., different layers and hidden sizes. Table 1 summarizes the models we investigate in this work[3]. All of these models are for English text and uncased.

For tokens that are broken into subwords by the tokenizer, we average the subword embeddings for the token representation. We use the models provided HuggingFace (Wolf et al., 2020), and Pytorch (Paszke et al., 2019) for all our experiments.

3.2 Tasks

We instantiate our analysis of the BERT models on a diverse set of four NLP tasks, which covers syntactic and semantic predictions. Here, we briefly describe the tasks, and refer the reader to the original sources of the data for further details.

Part-of-speech tagging (POS) predicts the part-of-speech tag for each word in a sentence. The task helps us understand if a representation captures coarse grained syntactic categorization. We use the English portion of the parallel universal dependencies treebank (ud-pud, Nivre et al., 2016).

Dependency relation (DEP) predicts the syntactic dependency relation between two tokens, i.e. (w_{head} and w_{mod}). This task can help us understand if, and how well, a representation can characterize syntactic relationships between words. Unlike the other tasks, this task involves assigning a category to a *pair* of tokens. We concatenate their contextualized representations from BERT and treat the concatenation as the representation of the pair. We use the same dataset as the POS task for dependencies.

Preposition supersense disambiguation involves the two categorization tasks of predicting **preposition's semantic role (PS-role)** and **semantic function (PS-fxn)**. Following the previous work (Liu et al., 2019), we only train and evaluate on single-token prepositions from Streusle v4.2 corpus (Schneider et al., 2018).

3.3 Fine-tuning Setup

We fine-tune all the representations in §3.1 on the four tasks from §3.2 separately. The fine-tuned models are then used to generate contextualized representations. The probing techniques described in §2 are applied to study these fine-tuned contextualized representations.

Our preliminary experiments found that the commonly used 3-5 epochs of fine-tuning do not show improvements for the smaller representations, such as $\text{BERT}_{\text{tiny}}$, and they require more epochs. We fine-tuned

[3]We ignore the $\text{BERT}_{\text{large}}$ because, during preliminary experiments, we found $\text{BERT}_{\text{large}}$ is highly unstable. The variance between different fine-tuning runs is so large that not comparable with other BERT models. This is consistent with the observations from Mosbach et al. (2020a)

all the representations for 10 epochs except $\text{BERT}_{\text{base}}$, which we fine-tuned for the usual five epochs. For each fine-tuned representation, we directly use the performance of the jointly learned classifier, which is trained during the fine-tuning process. To account for the instability of fine-tuning process (Lee et al., 2020; Dodge et al., 2020; Mosbach et al., 2020a), for each representation and task, we fine-tuned five times using different initializations and report the average accuracy and standard deviation.

4 Observations and Analysis

In this section, we will first see how well our representations encode the linguistic information by applying classifer probes (§4.1, §4.2), and then analyze the geometry of the representations using DIRECTPROBE (§4.3–§4.5).

4.1 Original Performances

Before analyzing the fine-tuning process, let us first compare the predictive performances for the original (i.e., before fine-tuning) representations of different sizes. We find the best performance for each original representation by training a two-layer neural network as described in §2.1.

Larger representations help semantic tasks more. Table 2 summarizes the performance of the five representations on the different tasks. As expected, the smaller representations usually have lower performance than the larger ones, indicating more parameters can encode more information. Across the different capacities, we observe that the improvements on semantic tasks (PS-role and PS-fxn) are larger than the syntax tasks (POS-tagging and Dependency). From $\text{BERT}_{\text{tiny}}$ to $\text{BERT}_{\text{base}}$, PS-fxn improves 13.61% ($71.14\% \rightarrow 87.75\%$) while POS-tagging only improves 2.62% ($90.76\% \rightarrow 93.39\%$). This large difference suggests that semantic tasks are harder than syntactic tasks. The former require representations with larger capacity.

4.2 Fine-tuned Performance

Now, we turn to analyze the fine-tuned performance. It is commonly accepted that the fine-tuning improves task performance. Here, we are interested in cases where this does not happen. Table 3 summarizes the interesting observations from our experiments. For the complete fine-tuning results, we refer the reader to the appendix.

Fine-tuning diverges the training and test set. In Table 3, the last column shows the spatial similarity (described in §2) between the training set and the test set for each representation. We can observe that after fine-tuning, all the similarities decrease, implying that the training and test sets become divergent as a result of fine-tuning.

Fine-tuning can memorize a training set in the representation parameters. An interesting observation in Table 3 is that $\text{BERT}_{\text{small}}$ and $\text{BERT}_{\text{medium}}$ do not show the improvements on the PS-fxn task after fine-tuning. We also observe $\text{BERT}_{\text{small}}$ has the smallest similarity between the training and test sets after fine-tuning for the task. Based on these observations, we hypothesize that fine-tuning can memorize the training set. To validate our hypothesis, we partition the training set into two 80:20 parts, called the *subtrain* and *subtest* sets respectively. The difference between the subtest and test set is that the subtest set is used during fine-tuning, while the test set is not. Only the subtrain set is used for training the classifier over a representation (either the original one, or fine-tuned). Table 4 summarizes the use of each dataset. We train a two-layer neural network on subtrain for the PS-fxn task and evaluate on the subtest and test sets. By

Representation	Task	Acc	Std
$\text{BERT}_{\text{tiny}}$	POS	90.76	0.24
	DEP	86.74	0.22
	PS-fxn	74.14	1.42
	PS-role	58.38	0.78
$\text{BERT}_{\text{mini}}$	POS	93.81	0.10
	DEP	91.82	0.00
	PS-fxn	82.45	1.07
	PS-role	68.05	1.08
$\text{BERT}_{\text{small}}$	POS	94.26	0.13
	DEP	92.93	0.14
	PS-fxn	86.26	0.54
	PS-role	74.22	1.03
$\text{BERT}_{\text{medium}}$	POS	94.40	0.08
	DEP	92.54	0.14
	PS-fxn	86.56	0.41
	PS-role	76.28	1.00
$\text{BERT}_{\text{base}}$	POS	93.39	0.31
	DEP	89.39	0.08
	PS-fxn	87.75	0.41
	PS-role	74.49	0.84

Table 2: Performances of the five representations on different tasks before fine-tuning. These results come from the last layer of each representation model.

comparing the *classifier* learning curves of subtest and test, we can verify if fine-tuning the representation memorizes the subtest.

Figure 2 shows the learning curves before and after fine-tuning. We can observe that before fine-tuning, subtest and test have a similar learning curve; while after fine-tuning, subtrain and subtest have a similar curve although subtest is not visible for classifier training. This means the *representation* memorizes the subtest during fine-tuning, such that subtrain and subtest share the same regularity. This phenomenon is not unique for the $\text{BERT}_{\text{small}}$-supersense function case. We see similar observations for other representations and tasks (shown in the appendix). Although this observation of memorization cannot explain why $\text{BERT}_{\text{small}}$ does not improve the performance after fine-tuning, it points to a possible direction for investigation: If the memorization becomes severe, will the fine-tuning process overfit the training set and can not generalize to unseen examples? We leave this question for future research.

4.3 Linearity of Representations

Next, let us examine the geometry of the representations before and after fine-tuning using DIRECTPROBE and counting the number of clusters[4].

Smaller representations require more complex classifiers. Table 5 summarizes the results. For brevity, we only present the results on $\text{BERT}_{\text{tiny}}$ and $\text{BERT}_{\text{mini}}$. All other representations are linear for all tasks; the full results are in the appendix. We observe that small representations (i.e., $\text{BERT}_{\text{tiny}}$ or $\text{BERT}_{\text{mini}}$) are non-linear for most tasks. Although a non-linearity does not imply poor generalization, it represents

[4]For the fine-tuned representations, we randomly pick one and apply the DIRECTPROBE.

Rep	Task		Acc	Sim
$\text{BERT}_{\text{small}}$	POS	originalal	94.25	0.96
		tuned	95.17	0.71
	DEP	original	92.93	0.92
		tuned	94.57	0.78
	PS-fxn	original	86.26	0.81
		tuned	*84.90*	**0.43**
	PS-role	original	74.22	0.83
		tuned	76.80	0.53
$\text{BERT}_{\text{medium}}$	POS	original	94.40	0.97
		tuned	95.34	0.66
	DEP	original	92.53	0.93
		tuned	95.16	0.79
	PS-fxn	original	86.56	0.79
		tuned	*86.12*	0.59
	PS-role	original	76.27	0.83
		tuned	78.29	0.58
$\text{BERT}_{\text{base}}$	POS	original	93.39	0.96
		tuned	95.44	0.69
	DEP	original	89.39	0.91
		tuned	94.72	0.75
	PS-fxn	original	87.74	0.83
		tuned	89.58	0.57
	PS-role	original	74.48	0.81
		tuned	82.05	0.52

Table 3: Fine-tuned performances of three BERT representations based on the last layers. The last column shows the spatial similarity (described in §2) between the training set and test set. A complete table of all representations and tasks can be found in the appendix.

a more complex spatial structure, and requires a more complex classifier. This suggests that to use small representations (say, due to limited resources), it would be advisable to use a non-linear classifier rather than a simple linear one.

Fine-tuning makes the space simpler. In Table 5, we observe that the number of clusters decreases after fine-tuning. This tells us that after fine-tuning, the points associated with different labels are in a simpler spatial organization. This is consistent with the common intuition that fine-tuning improves performance.

4.4 Spatial Structure of Labels

To fully understand the changes in spatial structure, we applied DIRECTPROBE to every intermediate representation encountered while fine-tuning. Here, we focus on the $\text{BERT}_{\text{base}}$.[5] Since all representations we considered are linearly separable, the number of clusters equals the number of labels. As a result, each cluster corresponds to a label uniquely and exclusively. Going ahead, we will use clusters and labels interchangeably.

[5]Due to computational constraints, in this experiment, we only fine-tuned three epochs for $\text{BERT}_{\text{base}}$.

	Visibility	
	fine-tuning	classifier training
subtrain	√	√
subtest	√	×
test	×	×

Table 4: The visibility of each subset for the fine-tuning and classifiers training. The representaion is fine-tuned on the union of subtrain and subtest while the classifier is only trained on sutrain set.

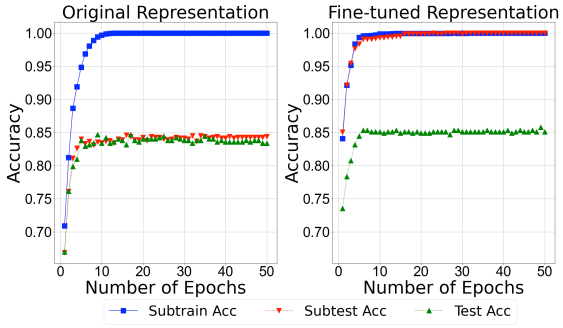


Figure 2: A learning curve comparison before and after fine-tuning on supersense function task. The horizontal axis is the number of epochs during classifier training; the vertical axis is the accuracy. The classifiers are trained on the last layer of $\text{BERT}_{\text{small}}$.

Fine-tuning pushes each label far away from each other. This confirms the observation of Zhou and Srikumar (2021), who pointed out that the fine-tuning pushes each label away from each other. However, they use the global minimum distance between clusters to support this argument, which only provides a partial support to the claim. The distancs between clusters might increase while the global minimum distanc decreases.

We track the minimum distance of each label to all other labels during fine-tuning. We find that all the minimum distances are increasing. Figure 3 shows the how these distances change in the last layer of $\text{BERT}_{\text{base}}$ for the supersense role and POS tagging tasks. The appendix includes the plots for all tasks. For clarity, we only show the three labels where the distance increases the most, and the three where it decreases the most. We also observe that although the trend is increasing, the minimum distance of each label can decrease during the course of fine-tuning, e.g., the label STUFF in supersense role task, suggesting the instability of fine-tuning.

To further show how the labels move during the fine-tuning , we compute the centroids of each cluster to represent the label position. We select three closest labels from the POS tagging task and track the paths of the centroids of each label cluster in the last layer of $\text{BERT}_{\text{base}}$ during the fine-tuning. Figure 4 (right) shows the PCA projection in 2D of these paths. We observe that before fine-tuning, the centroids of all these three labels are close to each other. As the fine-tuning proceeds, the centroids move around in different directions and become far away from each other.

We conclude that fine-tuning pushes each label away from others. This larger gap between labels admits more classifiers consistent with the labels, and allows for better generalization.

Representation	Task		#clusters	is linear
$\text{BERT}_{\text{tiny}}$	POS	original	3936	N
		tuned	20	N
	DEP	original	653	N
		tuned	46	Y
	PS-fxn	original	402	N
		tuned	40	Y
	PS-role	original	46	Y
		tuned	46	Y
$\text{BERT}_{\text{mini}}$	POS	original	2429	N
		tuned	17	Y
	DEP	original	46	Y
		tuned	46	Y
	PS-fxn	original	40	Y
		uned	40	Y
	PS-role	original	46	Y
		tuned	46	Y

Table 5: The linearity of the last layer of representation models for each task. This table only shows $\text{BERT}_{\text{tiny}}$ and $\text{BERT}_{\text{mini}}$, other results are in the appendix.

4.5 Layer Behavior

Previous work (Merchant et al., 2020; Mosbach et al., 2020b) showed that during fine-tuning, lower layers changed little compared to higher layers. In the following experiments, we verify their findings and show: (i) fine-tuning does not change the representation arbitrarily, even for higher layers; (ii) a quantitative analysis of the changes of different layers by a visual comparison between lower layers and higher layers. For simplicity, here, we focus on the POS tagging task. However, the same conclusions can be extended to other tasks. Experimental reuslts of other tasks are showed in the appendix.

Higher layers change more than the lower layers. First, we are interested in quantitatively analyzing the changes to different layers. We use each cluster's centroid to represent each label's position and quantify its movement by computing the Euclidean distance between the centroids before and after fine-tuning for each label, for all layers. Figure 5 summarizes the results. We can observe that as the layer increases, the distance becomes larger.

Higher layers do not change arbitrarily. Although Figure 5 shows that higher layers change more than the lower layers, we find that higher layers still remain close to the original representations. To study the dynamics of fine-tuning, we compared the intermediate representation of each layer during fine-tuning to its corresponding original pre-trained one. The similarity between two representations is calculated as the Pearson correlation coefficient of their distance vectors as described in §2.

Figure 6 shows the results for all four tasks. For simplicity, we only show the plots for every alternate layer. First, we observe that higher layers change more than the lower layers, and lower layers do not change during the fine-tuning. This observation is consistent with Figure 5. Second, even for the higher layers, we find that the Pearson correlation coefficient still shows strong linear relation (more than 0.5) between the original representation and the fine-tuned one, suggesting that fine-tuning does not change the representation

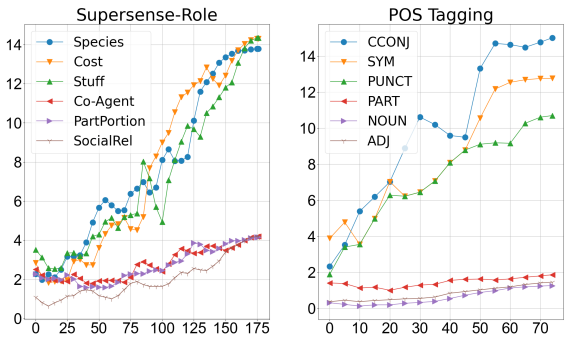


Figure 3: The dynamics of the minimum distances of the three labels where the distance increases the most, and the three where it decreases the most. The horizontal axis is the number of fine-tuning updates; the vertical axis is chosen label's minimum distance to other labels. These results come from the last layer of $\text{BERT}_{\text{base}}$. A full plots of four tasks can be found in the appendix.

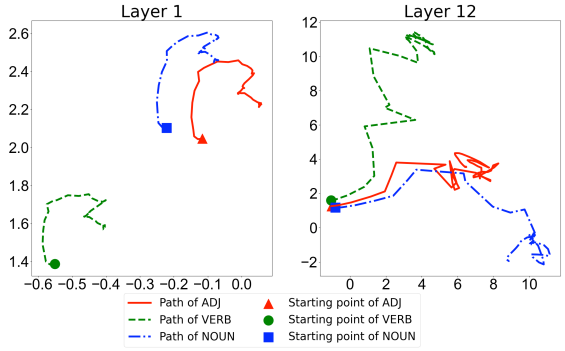


Figure 4: The PCA projection of three closest labels in POS tagging task based on the first (left) and last (right) layer of $\text{BERT}_{\text{base}}$. These lines show the paths of the centroids of each label cluster during the fine-tuning. The markers indicate the starting points. This figure is best seen in color.

arbitrarily. Instead, it pushes each label away from each other while preserving the relative positions of each label. This means the fine-tuning process encode the task-dependent information while preserving the pre-trained information as much as possible.

The labels of lower layers move only in a small region and almost in the same directions. The unchanged nature of lower layers raises the question: do these representations not change at all? To answer this question, for every label, we compute difference between its centroids before and after fine-tuning. Figure 7 shows the PCA projection in 2D of these difference vectors. For brevity, we only present the plots for every alternative layer. A plot with all layers can be found in the appendix. We observe that the movements of labels in lower layers concentrate in a few directions compared to the higher layers, suggesting the labels in lower layers do change, but do not separate the labels as much as the higher layers. Also, we observe that the labels INTJ and SYM have distinctive directions in the lower layers. Note that, in Figure 7, the motion range of lower layers is much smaller than the higher layers. The two projected dimensions range from -1 to 3 and from -3 to 3 for layer two, while for layer 12 they range from -12 to 13 and -12 to 8, suggesting that labels in lower layers only move in a small region compared to higher layers.

Figure 4 shows an example of this difference. Compared to the layer 12 (right) paths, we see that the layer 1 paths (left) traverse almost the same trajectories, which is consistent with the observations from Figure 7.

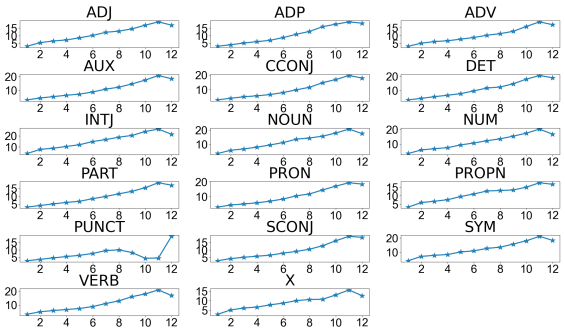


Figure 5: Euclidean distances between the centroids of labels before and after fine-tuning based on POS tagging task and $\text{BERT}_{\text{base}}$. The horizontal axis is the layer index; the vertical axis is the quantified movement, represented by Euclidean distances.

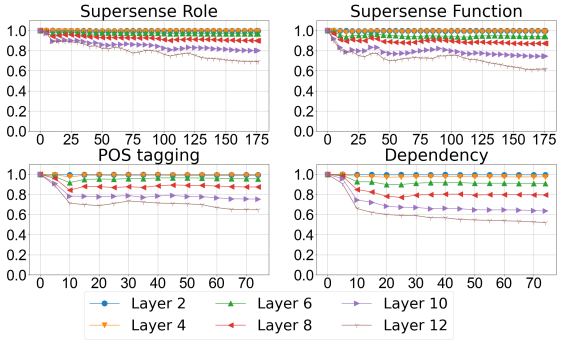


Figure 6: Dynamics of spatial similarity during the fine-tuning process based on $\text{BERT}_{\text{base}}$. The horizontal axis is the number of gradient updates during fine-tuning. The vertical axis is the Pearson correlation coefficient between current space and its original version (space before fine-tuning).

5 Related Work

There are many different lines of work that focus on analyzing and understanding representations. The most commonly used technique is the classifier-based method. Early work (Alain and Bengio, 2017; Kulmizev et al., 2020) starts with using linear classifiers to probe the representations. Hewitt and Liang (2019) pointed out that a simple linear probe is not enough to evaluate a representation. Recently, we have also seen non-linear probes (Tenney et al., 2019; Eger et al., 2019). There are also efforts to inspect the representations from a geometric persepctive (e.g. Ethayarajh, 2019; Mimno and Thompson, 2017), including the recently proposed DIRECTPROBE (Zhou and Srikumar, 2021), which we use in this work. Another line of probing work is to design control tasks (Ravichander et al., 2021; Lan et al., 2020) to reverse-engineer the internal mechanisms of representations (Kovaleva et al., 2019; Wu et al., 2020). However, in contrast to our work, most studies focused on the pre-trained representations, not the fine-tuned ones.

While fine-tuning pre-trained representations usually provides strong empirical performance (Wang et al., 2018; Talmor et al., 2020), how fine-tuning changes the representation to do so has remained an open question. Moreover, the instability (Mosbach et al., 2020a; Dodge et al., 2020; Zhang et al., 2020) and forgetting problems (Chen et al., 2020; He et al., 2021) of fine-tuning make it harder to analyze fine-tuned representations. With these difficulties, previous work (Merchant et al., 2020; Mosbach et al., 2020b; Hao et al., 2020) draw valuable conclusions about fine-tuning. This work extends this line of effort and provides a deeper understanding of how fine-tuning changes the representations

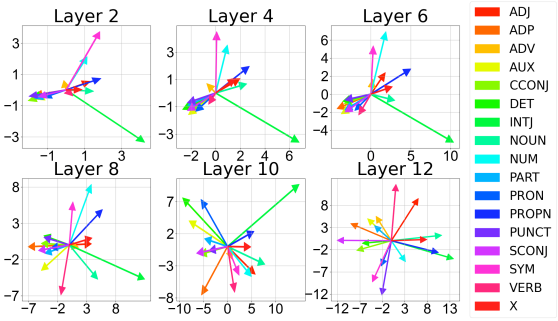


Figure 7: The PCA projection of the difference vector between the centroids of labels before and after fine-tuning based on POS tagging task and $\text{BERT}_{\text{base}}$. Lower layers have a much smaller projection range than the higher layers. This figure is best seen in color.

6 Conclusions

In this work, we take a close look at what fine-tuning does to contextualized representations. We investigate the fine-tuned representations of several BERT models using two probing techniques: classifier-based probing and DIRECTPROBE. Our experiments suggest that fine-tuning pushes the points associated with a label away from other labels, which leads to the improvements of task performance. We also discover that fine-tuning adjusts the representations to downstream tasks while preserving the original geometric structure of the points as much as possible. With carefully designed experiments, we show that a training set can be encoded in the representations by the fine-tuning, suggesting a potential risk of a new kind of overfitting.

References

Samira Abnar, Lisa Beinborn, Rochelle Choenni, and Willem Zuidema. 2019. Blackbox meets blackbox: Representational similarity & stability analysis of neural language models and brains. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 191–203, Florence, Italy. Association for Computational Linguistics.

Guillaume Alain and Yoshua Bengio. 2017. Understanding intermediate layers using linear classifier probes. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27.

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7870–7881, Online. Association for Computational Linguistics.

Grzegorz Chrupała and Afra Alishahi. 2019. Correlating neural and symbolic representations of language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2952–2962, Florence, Italy. Association for Computational Linguistics.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings*

of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah A. Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *CoRR*, abs/2002.06305.

Steffen Eger, Andreas Rücklé, and Iryna Gurevych. 2019. Pitfalls in the evaluation of sentence embeddings. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 55–60, Florence, Italy. Association for Computational Linguistics.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China. Association for Computational Linguistics.

Jon Gauthier and Roger Levy. 2019. Linking artificial and human neural representations of language. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 529–539, Hong Kong, China. Association for Computational Linguistics.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2020. Investigating learning dynamics of BERT fine-tuning. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 87–92, Suzhou, China. Association for Computational Linguistics.

Tianxing He, Jun Liu, Kyunghyun Cho, Myle Ott, Bing Liu, James Glass, and Fuchun Peng. 2021. Analyzing the forgetting problem in pretrain-finetuning of open-domain dialogue response models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1121–1133, Online. Association for Computational Linguistics.

John Hewitt and Percy Liang. 2019. Designing and interpreting probes with control tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2733–2743, Hong Kong, China. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Nora Kassner and Hinrich Schütze. 2020. Negated and misprimed probes for pretrained language models: Birds can talk, but cannot fly. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7811–7818, Online. Association for Computational Linguistics.

Najoung Kim, Roma Patel, Adam Poliak, Patrick Xia, Alex Wang, Tom McCoy, Ian Tenney, Alexis Ross, Tal Linzen, Benjamin Van Durme, Samuel R. Bowman, and Ellie Pavlick. 2019. Probing what different NLP tasks teach machines about function word comprehension. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 235–249, Minneapolis, Minnesota. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, Hong Kong, China. Association for Computational Linguistics.

Katarzyna Krasnowska-Kieraś and Alina Wróblewska. 2019. Empirical linguistic study of sentence embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5729–5739, Florence, Italy. Association for Computational Linguistics.

Artur Kulmizev, Vinit Ravishankar, Mostafa Abdou, and Joakim Nivre. 2020. Do neural language models show preferences for syntactic formalisms? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4077–4091, Online. Association for Computational Linguistics.

Aarre Laakso and Garrison Cottrell. 2000. Content and cluster analysis: assessing representational similarity in neural systems. *Philosophical psychology*, 13(1):47–76.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2020. Mixout: Effective regularization to finetune large-scale pretrained language models. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019. Linguistic knowledge and transferability of contextual representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Amil Merchant, Elahe Rahimtoroghi, Ellie Pavlick, and Ian Tenney. 2020. What happens to BERT embeddings during fine-tuning? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 33–44, Online. Association for Computational Linguistics.

David Mimno and Laure Thompson. 2017. The strange geometry of skip-gram with negative sampling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2873–2878, Copenhagen, Denmark. Association for Computational Linguistics.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020a. On the stability of fine-tuning BERT: misconceptions, explanations, and strong baselines. *CoRR*, abs/2006.04884.

Marius Mosbach, Anna Khokhlova, Michael A. Hedderich, and Dietrich Klakow. 2020b. On the interplay between fine-tuning and sentence-level probing for linguistic knowledge in pre-trained transformers. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 68–82, Online. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Christian S Perone, Roberto Silveira, and Thomas S Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259*.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 7–14, Florence, Italy. Association for Computational Linguistics.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5231–5247, Online. Association for Computational Linguistics.

Abhilasha Ravichander, Yonatan Belinkov, and Eduard Hovy. 2021. Probing the probing paradigm: Does probing accuracy entail task relevance? In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3363–3377, Online. Association for Computational Linguistics.

Nathan Schneider, Jena D. Hwang, Vivek Srikumar, Jakob Prange, Austin Blodgett, Sarah R. Moeller, Aviram Stern, Adi Bitan, and Omri Abend. 2018. Comprehensive supersense disambiguation of English prepositions and possessives. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 185–196, Melbourne, Australia. Association for Computational Linguistics.

Alon Talmor, Yanai Elazar, Yoav Goldberg, and Jonathan Berant. 2020. olmpics - on what language model pre-training captures. *Trans. Assoc. Comput. Linguistics*, 8:743–758.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? probing numeracy in embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5307–5315, Hong Kong, China. Association for Computational Linguistics.

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, Brussels, Belgium. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

Zhiyong Wu, Yun Chen, Ben Kao, and Qun Liu. 2020. Perturbed masking: Parameter-free probing for analyzing and interpreting BERT. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4166–4176, Online. Association for Computational Linguistics.

Yadollah Yaghoobzadeh, Katharina Kann, T. J. Hazen, Eneko Agirre, and Hinrich Schütze. 2019. Probing for semantic classes: Diagnosing the meaning content of word embeddings. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5740–5753, Florence, Italy. Association for Computational Linguistics.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2020. Revisiting few-sample bert fine-tuning. *arXiv preprint arXiv:2006.05987*.

Yichu Zhou and Vivek Srikumar. 2021. DirectProbe: Studying representations without classifiers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5070–5083, Online. Association for Computational Linguistics.

A Probing Performance

Table 6 shows the complete table of probing results in our experiments. The last column is the spatial similarity between the training set and test set. Some entries are missing because the similarity can only be computed on the representations that are linearly separable for the given task.

B Euclidean Distances across Layers

Figures 8–11 show the Euclidean distances between the centroids of labels before and after fine-tuning for all four tasks using $\text{BERT}_{\text{base}}$.

Representations	Task		Acc	Std	#Cluster	is Linear	Similarity
$BERT_{tiny}$	POS	original	90.76	0.24	3936	N	-
		fine-tuned	92.05	0.32	20	N	-
	DEP	original	86.74	0.22	653	N	-
		fine-tuned	87.34	0.23	46	Y	0.88
	PS-fxn	original	74.14	1.42	402	N	-
		fine-tuned	76.19	0.89	40	Y	0.72
	PS-role	original	58.38	0.78	46	Y	0.76
		fine-tuned	62.19	1.60	46	Y	0.70
$BERT_{mini}$	POS	original	93.81	0.10	2429	N	-
		fine-tuned	94.70	0.13	17	Y	0.70
	DEP	original	91.82	0.09	46	Y	0.93
		fine-tuned	93.63	0.12	46	Y	0.86
	PS-fxn	original	82.45	1.07	40	Y	0.77
		fine-tuned	83.89	0.94	40	Y	0.53
	PS-role	original	68.05	1.08	46	Y	0.81
		fine-tuned	72.95	1.23	46	Y	0.59
$BERT_{small}$	POS	original	94.26	0.13	17	Y	0.96
		fine-tuned	95.17	0.16	17	Y	0.72
	DEP	original	92.93	0.14	46	Y	0.93
		fine-tuned	94.57	0.21	46	Y	0.78
	PS-fxn	original	86.26	0.54	40	Y	0.82
		fine-tuned	84.90	0.42	40	Y	0.44
	PS-role	original	74.22	1.03	46	Y	0.84
		fine-tuned	76.81	1.06	46	Y	0.54
$BERT_{medium}$	POS	original	94.40	0.08	17	Y	0.97
		fine-tuned	95.35	0.11	17	Y	0.67
	DEP	original	92.54	0.14	46	Y	0.94
		fine-tuned	95.16	0.24	46	Y	0.79
	PS-fxn	original	86.56	0.41	40	Y	0.80
		fine-tuned	86.13	0.71	40	Y	0.59
	PS-role	original	76.28	1.00	46	Y	0.83
		fine-tuned	78.29	1.34	46	Y	0.58
$BERT_{base}$	POS	original	93.39	0.31	17	Y	0.97
		fine-tuned	95.44	0.22	17	Y	0.70
	DEP	original	89.39	0.08	46	Y	0.92
		fine-tuned	94.73	0.22	46	Y	0.76
	PS-fxn	original	87.75	0.41	40	Y	0.84
		fine-tuned	89.58	0.67	40	Y	0.57
	PS-role	original	74.49	0.84	46	Y	0.82
		fine-tuned	82.06	1.30	46	Y	0.52

Table 6: A complete table of the probing results of five representations on four tasks.

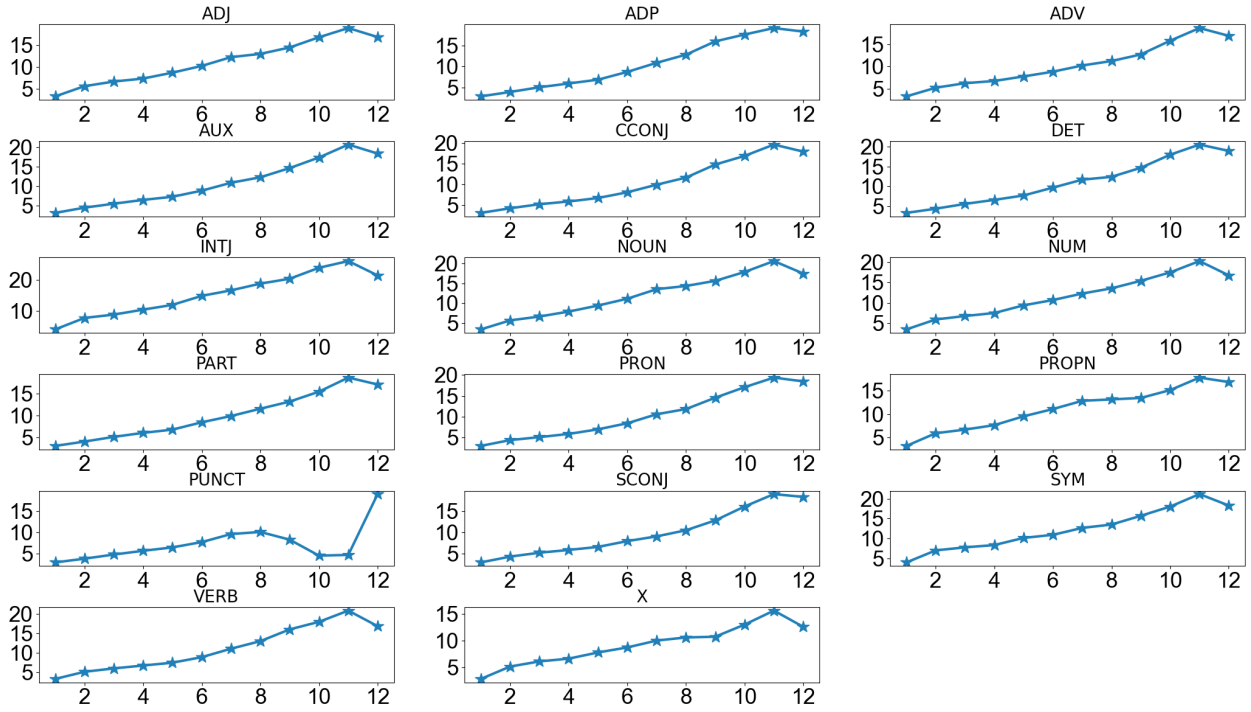


Figure 8: Euclidean distances between the centroids of labels before and after fine-tuning based on POS tagging task and $\text{BERT}_{\text{base}}$. The horizontal axis is the layer index; the vertical axis is the quantified movement, represented by Euclidean distances.

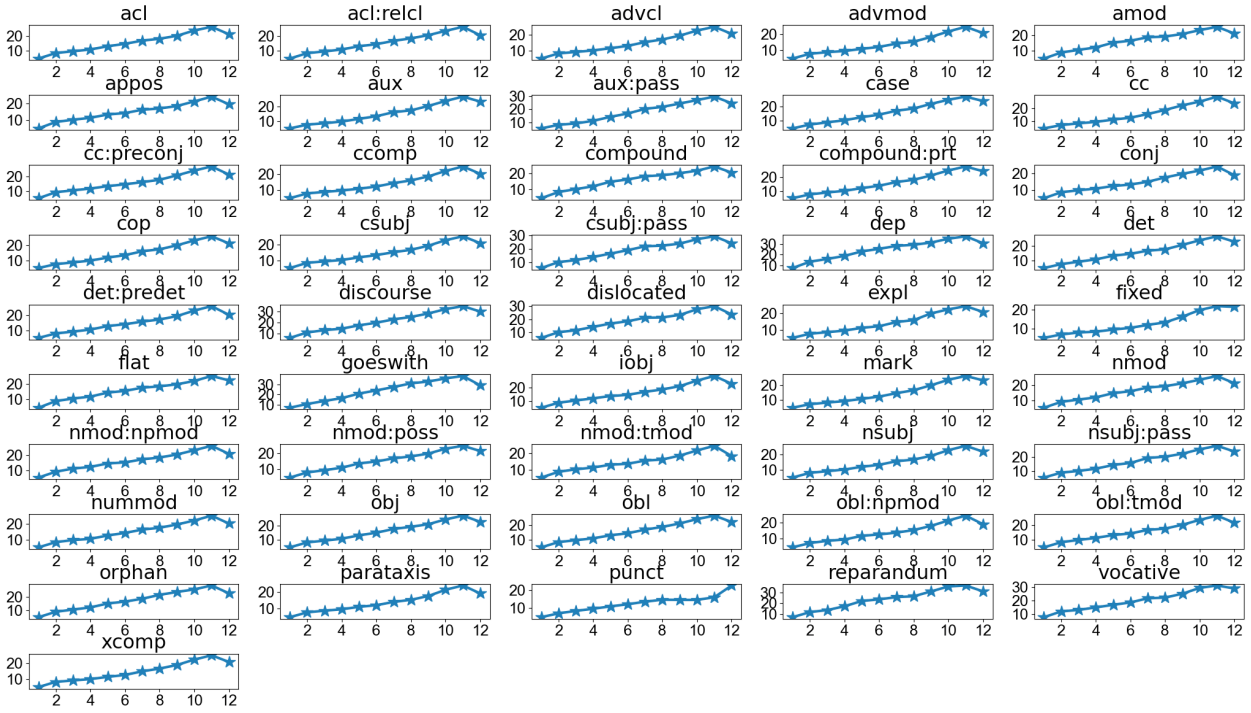


Figure 9: Euclidean distances between the centroids of labels before and after fine-tuning based on dependency relation prediction task and $\text{BERT}_{\text{base}}$. The horizontal axis is the layer index; the vertical axis is the quantified movement, represented by Euclidean distances.

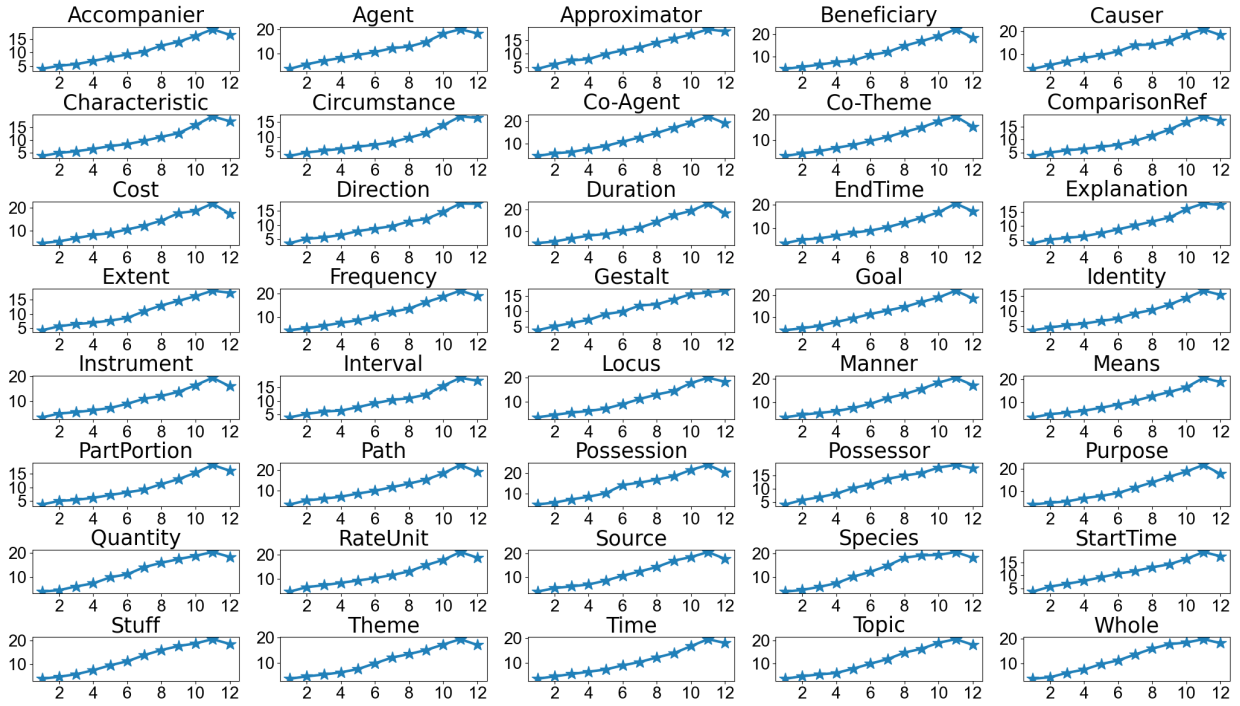


Figure 10: Euclidean distances between the centroids of labels before and after fine-tuning based on Supersense func prediction task and $\text{BERT}_{\text{base}}$. The horizontal axis is the layer index; the vertical axis is the quantified movement, represented by Euclidean distances.

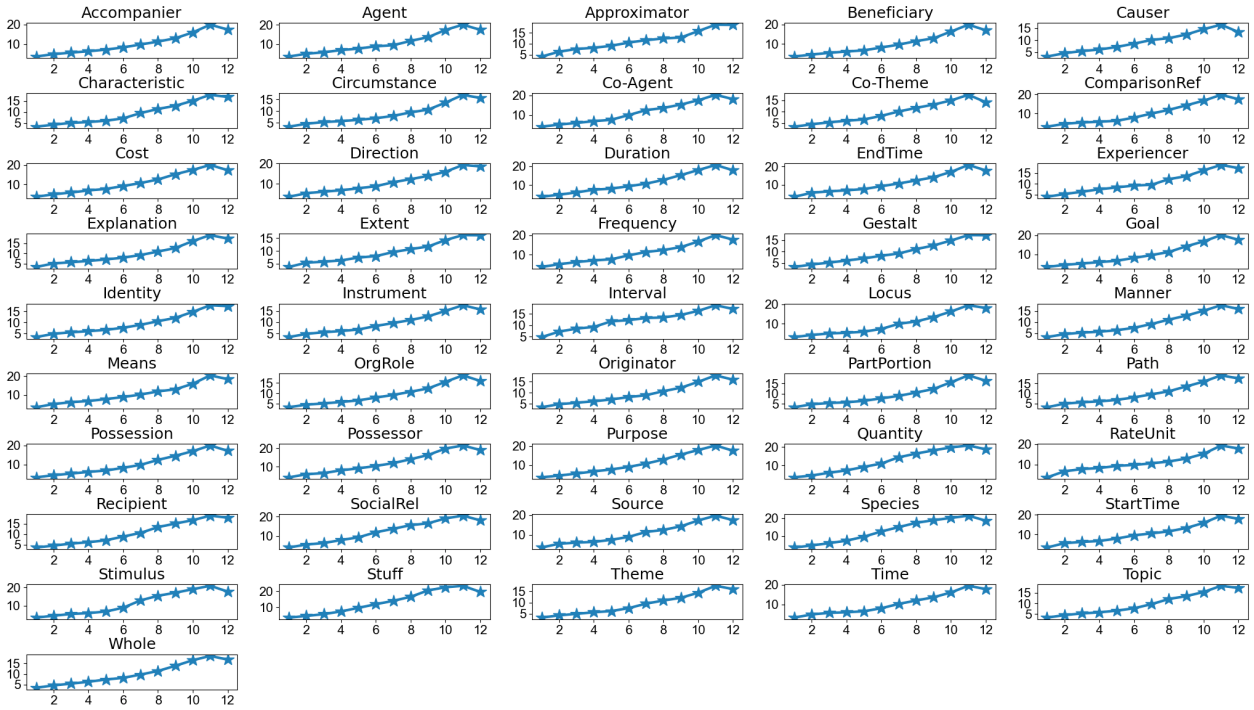
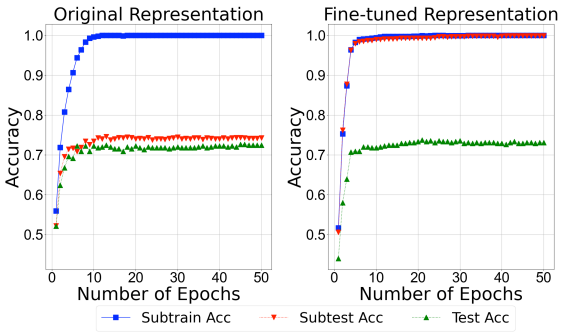
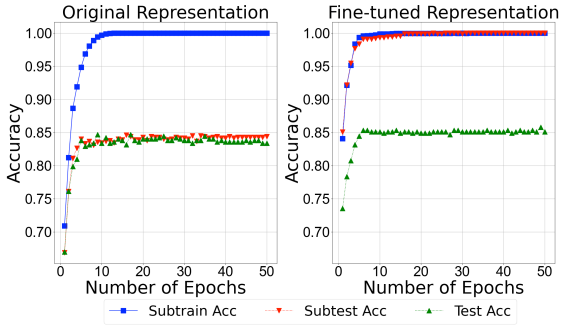


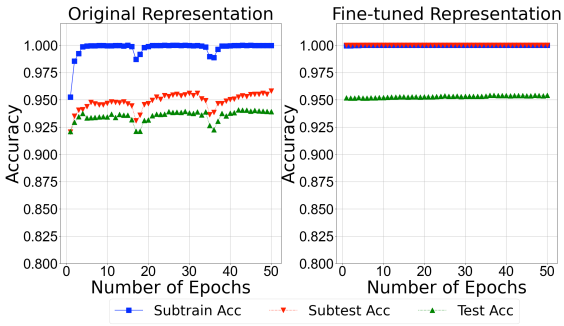
Figure 11: Euclidean distances between the centroids of labels before and after fine-tuning based on Supersense role prediction task and $\text{BERT}_{\text{base}}$. The horizontal axis is the layer index; the vertical axis is the quantified movement, represented by Euclidean distances.



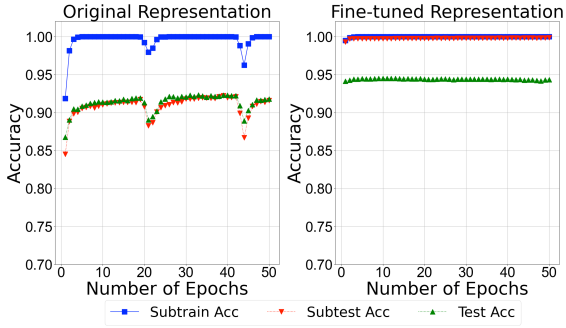
(a) Supersense role task.



(b) Supersense function task.



(c) POS tagging task.



(d) Dependency task.

Figure 12: Learning curve comparsions before and after fine-tuning for all four tasks on $\text{BERT}_{\text{small}}$. The horizontal axis is the number of epochs during classifier training; the vertical axis is the accuracy.

C Learning Curves

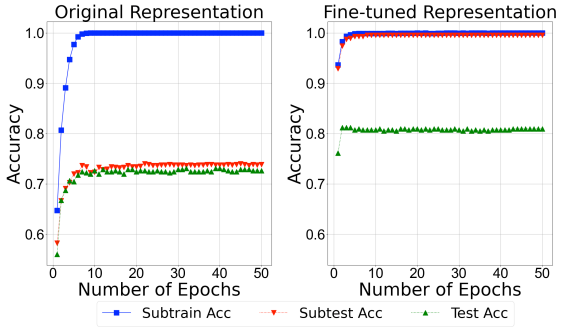
Figure 12 and Figure 13 show the learning curve comparsions before and after fine-tuning for all four tasks on $\text{BERT}_{\text{small}}$ and $\text{BERT}_{\text{base}}$. A two-layer neural network is trained on the subtrain set and evaluate on subtest and test set. The subtest is visible to the BERT models during fine-tuning while *not* visible for training classifiers. The test set is an unseen set for both fine-tuning and classifier training.

D Dynamics of Minimum Distances

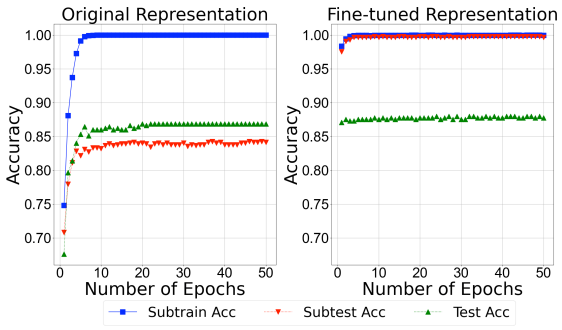
Figure 14 shows the dynamics of minimum distances for labels on all four tasks. For clarity, we only present the distances for the three labels where the distances increase the most and the three where it decreases the most.

E PCA Projections of the Movements

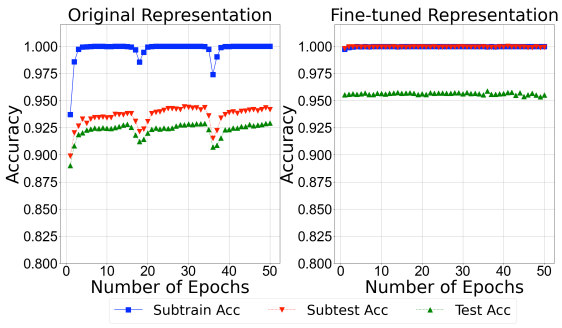
Figures 15–18 show the PCA projections of the difference vector between the centroids of labels before and after fine-tuning based on $\text{BERT}_{\text{base}}$.



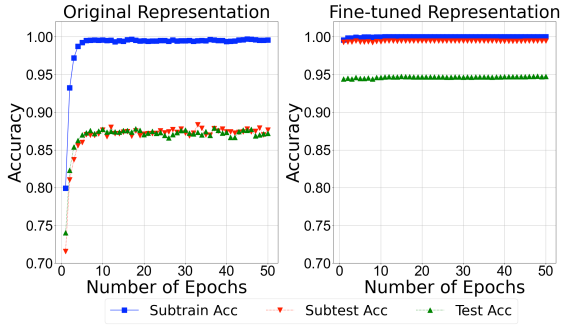
(a) Supersense role task.



(b) Supersense function task.



(c) POS tagging task.



(d) Dependency task.

Figure 13: Learning curve comparsions before and after fine-tuning for all four tasks on $\text{BERT}_{\text{base}}$. The horizontal axis is the number of epochs during classifier training; the vertical axis is the accuracy.

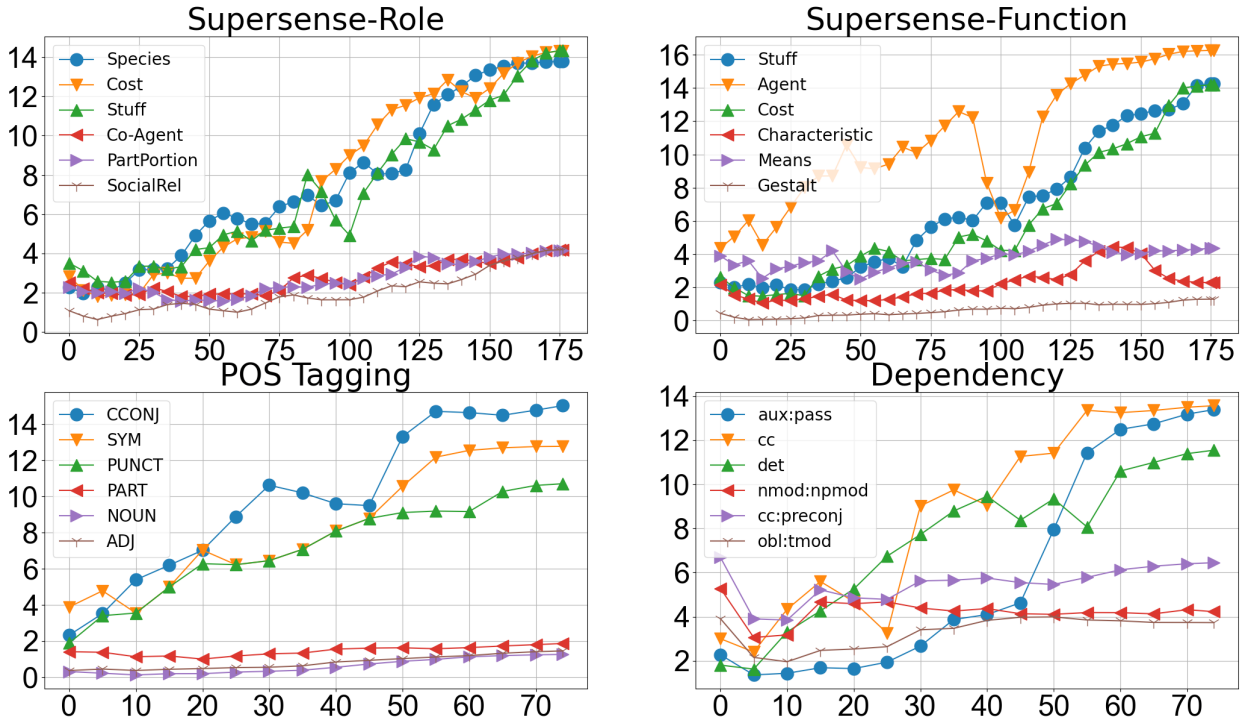


Figure 14: The dynamics of the minimum distance of the three most increased and three least increased labels. The horizontal axis is the number of fine-tuning updates; the vertical axis is chosen label's minimum distance to other labels. These results come from the last layer of $\text{BERT}_{\text{base}}$.

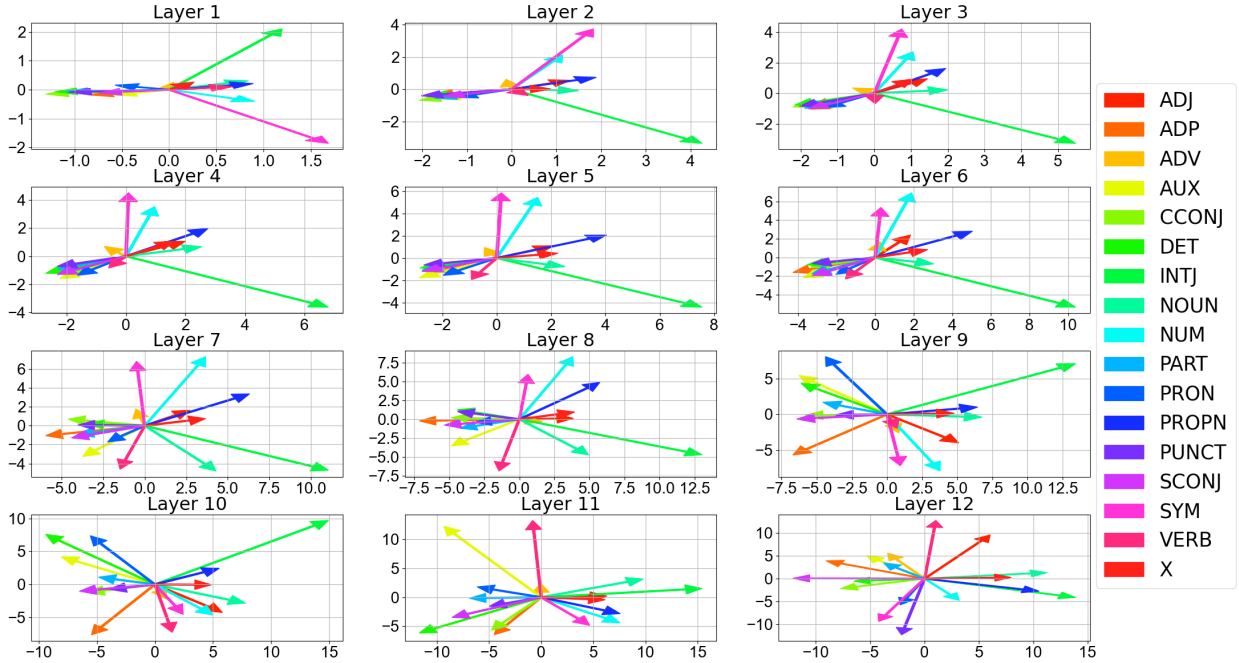


Figure 15: The PCA projection of the difference vector between the centroids of labels before and after fine-tuning based on POS tagging task and $\text{BERT}_{\text{base}}$.

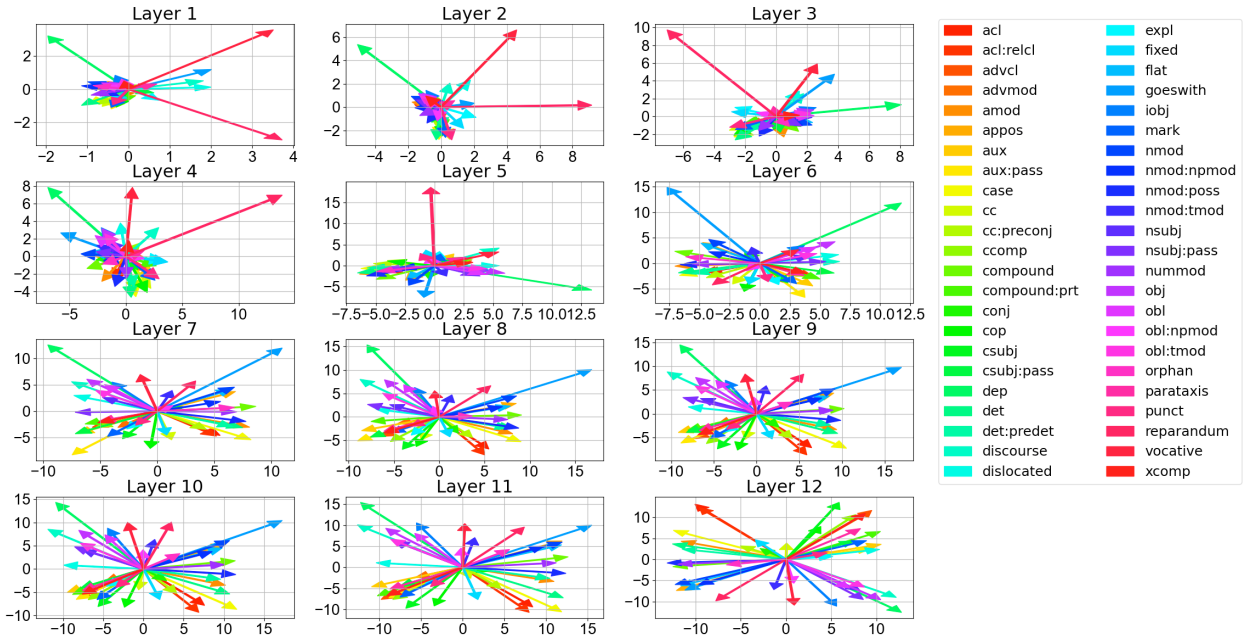


Figure 16: The PCA projection of the difference vector between the centroids of labels before and after fine-tuning based on dependency prediction task and $\text{BERT}_{\text{base}}$.

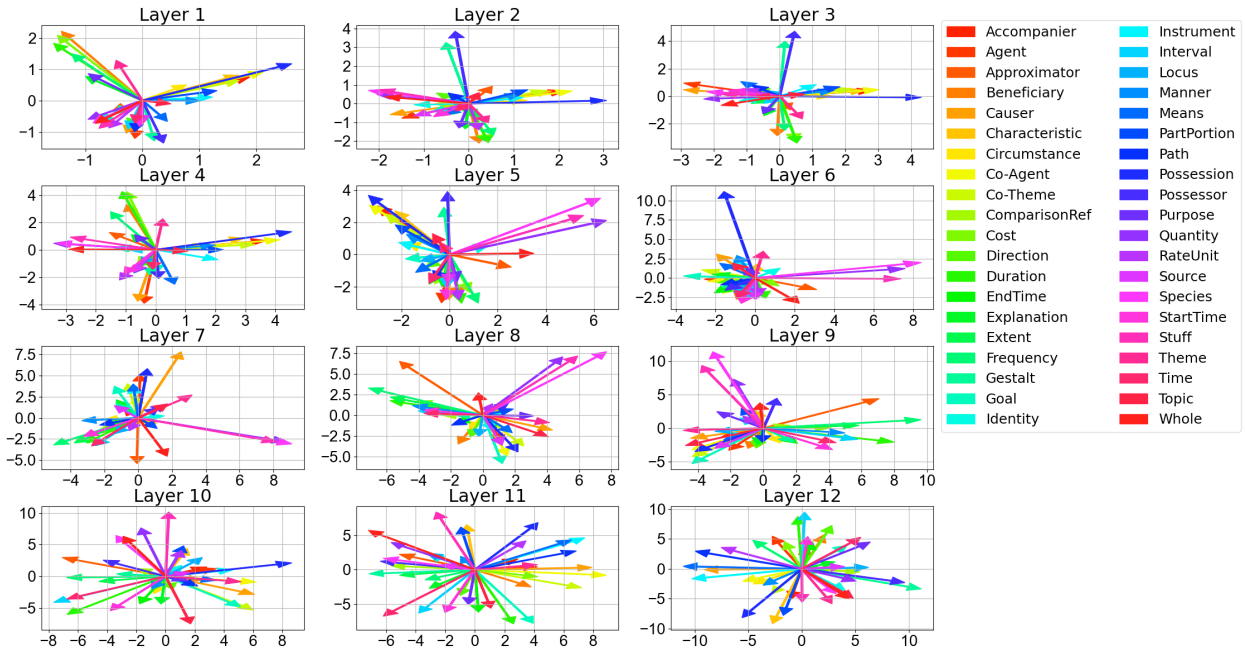


Figure 17: The PCA projection of the difference vector between the centroids of labels before and after fine-tuning based on Supersense function task and $\text{BERT}_{\text{base}}$.

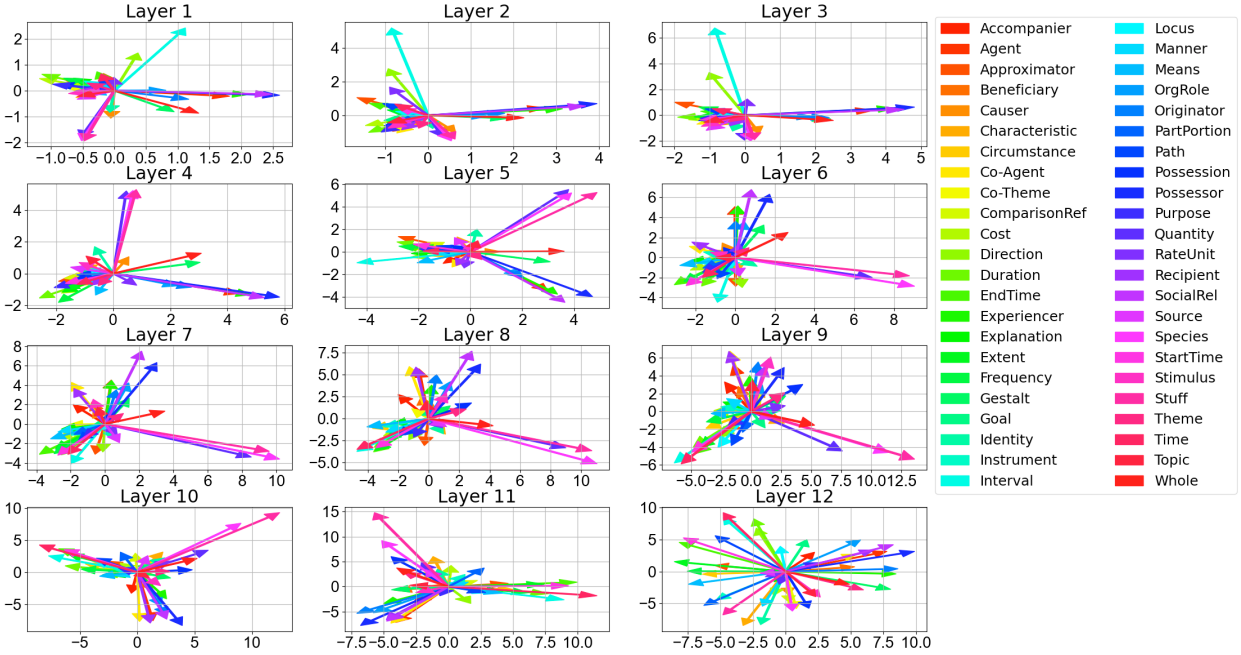


Figure 18: The PCA projection of the difference vector between the centroids of labels before and after fine-tuning based on Supersense role task and $\text{BERT}_{\text{base}}$.