On Attention Redundancy: A Comprehensive Study

Yuchen Bian, Jiaji Huang, Xingyu Cai, Jiahong Yuan, Kenneth Church
Baidu Research, Sunnyvale, CA, USA
{yuchenbian,huangjiaji,xingyucai,jiahongyuan,kennethchurch}@baidu.com

Abstract

Multi-layer multi-head self-attention mechanism is widely applied in modern neural language models. Attention redundancy has been observed among attention heads but has not been deeply studied in the literature. Using BERT-base model as an example, this paper provides a comprehensive study on attention redundancy which is helpful for model interpretation and model compression. We analyze the attention redundancy with Five-Ws and How. (What) We define and focus the study on redundancy matrices generated from pre-trained and fine-tuned BERT-base model for GLUE datasets. (How) We use both token-based and sentence-based distance functions to measure the redundancy. (Where) Clear and similar redundancy patterns (cluster structure) are observed among attention heads. (When) Redundancy patterns are similar in both pre-training and fine-tuning phases. (Who) We discover that redundancy patterns are task-agnostic. Similar redundancy patterns even exist for randomly generated token sequences. ("Why") We also evaluate influences of the pre-training dropout ratios on attention redundancy. Based on the phase-independent and task-agnostic attention redundancy patterns, we propose a simple zero-shot pruning method as a case study. Experiments on fine-tuning GLUE tasks verify its effectiveness. The comprehensive analyses on attention redundancy make model understanding and zero-shot model pruning promising.

1 Introduction

Multi-layer multi-head self-attention architectures (Transformer (Vaswani et al., 2017)) are widely applied in modern language models, such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), OpenAI GPT (Radford et al., 2018), GPT-2 (Radford et al., 2019) and ERNIE2.0 (Sun et al., 2019), to name a few.

Redundancy phenomenon is discovered among attention heads. It demonstrates that many attention heads generate very similar attention matrices (Clark et al., 2019; Kovaleva et al., 2019). We take the pre-trained BERT-base model as an example. It learns 12-layer-12-head self-attention matrices describing dependencies between each pair of tokens in a sentence. Then for each token, there are 144 attention vectors. We use Jensen-Shannon distance to measure the relationship between each pair of vectors. Then for one sentence (consisting of a sequence of tokens), the token-averaged distance is utilized to imply the redundancy between each pair of attention matrices. Smaller distance values reflect more redundancy. Figure 1 shows the redundancy (distance) among 144×144 pairs of attention matrices averaged over 1000 randomly sampled sentences. We can see clear redundancy patterns (clusters with smaller distance areas) in consecutive attention layers.

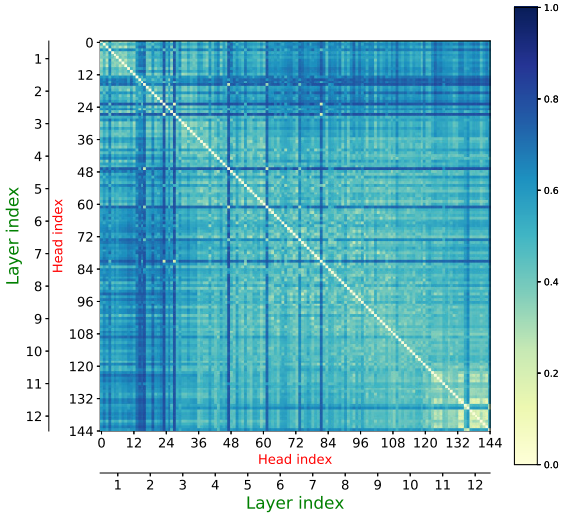


Figure 1: Pair-wise Jensen-Shannon distance of attention heads for the pre-trained BERT-base model (12-layer-12-head self-attention). Attention redundancy (cluster with small distances) exists in adjacent attention heads and layers.

Analyzing the attention redundancy helps to interpret the multi-layer multi-head self-attention architecture. Various studies have attempted to re-

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 930–945
June 6–11, 2021.

veal the relationship among attention heads. Examples are attention visualization (Vig and Belinkov, 2019), common attention patterns (Kovaleva et al., 2019), attention head pruning (Voita et al., 2019), and probing test (Clark et al., 2019). Existing works either focus on the 12×12 attention matrices and their effects on (pre-training or/and fine-tuning) performances or focus on linguistic features extracted by latent token vectors and attention matrices.

Though the redundancy phenomenon was discovered, no existing work studies the attention redundancy pattern itself (i.e., the 144×144 distance matrix in Figure 1) deeply. This motivates us to conduct a comprehensive and complementary study on the attention redundancy phenomenon.

In this paper, we take the BERT-base model as a representative model to analyze the attention redundancy with Five Ws and How. As far as we know, many of the following discoveries are new to the research community.

What is attention redundancy?

Given a distance function, we define the pairwise distance matrix ($\in \mathbb{R}^{144\times144}$) of the 12×12 attention matrices of BERT-base model as *attention redundancy matrix*. In this paper, we obtain redundancy matrices from both pre-trained and fine-tuned BERT-base model for GLUE tasks as the research objects.

How to measure attention redundancy?

Except for the two token-based measures, Jensen-Shannon distance (Clark et al., 2019) and cosine similarity (Kovaleva et al., 2019) used in literature, we employ two more token-based distance function and three sentence-based ones to measure attention redundancy and analyze their similar redundancy patterns (please refer to Section 4.1 for more details). The purpose is to alleviate the measuring bias of just using one distance function. Sentence-based distances directly measure the relationship between two attention matrices without averaging over tokens. We visualize the redundancy patterns using various distance functions.

Where does attention redundancy exist?

We find common hierarchical cluster structures in the set of token-based redundancy matrices and the set of sentence-based redundancy matrices, respectively. Attention heads of earlier, middle, and deeper attention layers are clearly clustered in the redundancy matrices. We also demonstrate that highly correlated similar redundancy patterns exist in redundancy matrices generated based on different type of distances.

When does attention redundancy occur?

The redundancy is phase-independent. Common redundancy patterns are discovered in both the pre-trained phase and fine-tuned phases. For any downstream task with any distance function, we notice highly correlated attention redundancy patterns between two phases.

Who (which task) has attention redundancy?

We surprisingly realize that the redundancy is task-agnostic. The redundancy patterns are highly correlated across different tasks. We even randomly generate token sequences as input in the pre-trained BERT-base model. Very similar attention redundancy patterns occur as well.

Based on this astonishing discovery, as a case study application, we propose a simple zero-shot head-pruning strategy based on clustering results using redundancy matrices. Compared to other complex pruning strategies, e.g., (Tang et al., 2019; Jiao et al., 2019; Fan et al., 2019; Wang et al., 2019; McCarley, 2019), the most important is that without knowing any data of fine-tuning tasks, this pruning can be effectively and efficiently conducted just based on some randomly generated token sequences with the pre-trained BERT-base model. The only effort is to compute one or several redundancy matrices. Results reflect that for most GLUE tasks, the proposed pruning strategy based on redundancy matrices can prune up to 75% to 85% of attention heads while keeping comparable fine-tuning performances.

"Why" does the phase-independent and task-agnostic attention redundancy happen?

It's hard to tell the reason of the redundancy patterns (that's why we use the quoted "Why"). However, we conduct experiments to evaluate the effects on attention redundancy of dropout ratios in the pre-training phase which are suspected as one of reasons (Clark et al., 2019).

When we use sentence-based distance, a monotonic trend is found. Attention heads tend to be more redundant when increasing dropout ratios. When we use token-based distances, a complex "N"-shape effect exists. We also notice that the redundancy is more sensitive to dropouts in hidden linear transformations than to dropouts in the self-attention mechanism.

We believe that above-mentioned new findings in this paper make the redundancy analyses a promis-

ing research direction in model interpretation and model compression, and so on.

2 Related Work

Existing literature analyzes multi-layer multi-head self-attention architectures based language models (e.g., BERT) from different aspects including word/token embedding latent space, linguistic knowledge interpretability, attention mechanism, and so on (Rogers et al., 2020).

The output of BERT attention layers are token embedding vectors. One output vector for one token aggregates contextual information from the whole sentence. In the vector space, attention can produce strong representations for syntactic phenomena and phrasal information (Jawahar et al., 2019) , but small improvements on semantic tasks (Tenney et al., 2019). Ethayarajh (2019) showed that contextualized representations of all words are not isotropic in any layer. Cai et al. (2021) revealed isotropy in the clustered contextual embedding space, and found low-dimensional manifolds. In the fine-tuning process, Kovaleva et al. (2019) showed that the last two attention layers encode task-specific features while earlier layers capture more fundamental information.

Another set of works focus on the ability of extracting linguistic knowledge. BERT can obtain syntactic dependencies, parts of speech tags, word disambiguation, and so forth (Ethayarajh, 2019; Vig and Belinkov, 2019; Clark et al., 2019; Goldberg, 2019). It showed that the same layer learns similar knowledge (Clark et al., 2019). Positional information is encoded in BERT lower layers (Lin et al., 2019). Vig and Belinkov (2019) argued that middle and last attention layers can extract dependency relations and distant information, respectively. Even with BERT's success, it still struggles handling some linguistic information and tasks. BERT does not excel at numbers, negation, inferences and role-based event prediction (Wallace et al., 2019; Ettinger, 2020). It's questionable to provide transparency or meaningful explanations for model predictions on downtream tasks (Jain and Wallace, 2019) .

Since self-attention is the fundamental mechanism in BERT, existing works also investigate extracted attention vectors and/or matrices. These are most relevant to our study. Clark et al. (2019) and Kovaleva et al. (2019) found some common attention patterns, such as patterns on delimiter tokens, block, and heterogeneous patterns. Also, redundancy and overparameterization is discovered in attention heads. By using attention-based probing classifiers, Clark et al. (2019) showed that heads in the same layer often exhibits similar behaviors. Attention heads can be pruned with different strategies but keep comparable performance in downstream tasks (Voita et al., 2019; Clark et al., 2019). Some layers can even be reduced to a single head (Michel et al., 2019).

Understanding attention redundancy can help interpret pre-trained/fine-tuned language models and guide model compression. But no systematic study on attention redundancy exists. This paper provides a deep study (Five-Ws and How) on the attention redundancy.

3 What: Redundancy Matrices

In this paper, we use BERT-base model as a representative to study the attention redundancy existing in multi-layer multi-head self-attention mechanisms. As a pre-trained language model, BERT has been verified with outstanding fine-tuning performances on many downstream language understanding tasks.

In BERT-base model there are 12 self-attention layers each of which consists of 12 self-attention heads. For one input (sentence or sentence pair with special tokens [CLS] and [SEP]), we extract 12×12 attention matrices. The size of each matrix is $n \times n$ where n is the number of tokens after BERT tokenization. Given a metric (e.g., distance function) and extracted attention matrices, we can measure the relationship between each pair of the 144 attention matrices, for instance the 144×144 matrix in Figure 1. We define the pair-wise relationship among the 144 heads (i.e., attention matrices) as *redundancy matrix*. We consider that the smaller the distances among some attention heads are, the more *attention redundancy* exists in them.

As for the studying objects, we use the well-known natural language understanding benchmark GLUE task[1] (Wang et al., 2018) as evaluation objects for the attention redundancy analysis. Among GLUE, CoLA is for English sentence acceptability judgments. SST-2 and STS-B are sentiment analyses tasks. MRPC and QQP are for sentence-pair similarity classification. MNLI, QNLI, and

[1]We eliminate WNLI due to its very small size. The same elimination was conducted in the literature (Kovaleva et al., 2019).

RTE are natural language inference tasks. CoLA and SST-2 are single sentence tasks, while others are predicting relationship between a pair of sentences. Except that STS-B is a regression task, in the BERT framework, others are formed as classification tasks.

In experiments, we randomly select 1000 data samples from each development set[2] as data instances and report the averaged results. We feed them to (pre-trained and fine-tuned) BERT-base model to generate attention matrices.

We use PyTorch BERT-base model as the pre-trained model[3] (Wolf et al., 2019). For fine-tuning, we train each task with suggested hyper-parameters *max-length*=128, *num-epoch*=3, *batch-size-per-GPU*=32 on 8 GPUs. All results in this paper are averaged over 5 trials.

4 How: Distance Functions

A fundamental question about attention redundancy is how to measure the similarity or distance between two attention matrices. The smaller the distance is, the more redundancy should exist. In this section, we introduce two levels of distance functions, token-based and sentence-based distances. The aim is to inspect their interpretabilities and consistency on quantifying the attention redundancy and to alleviate measuring bias of just using a certain distance function.

Let $dist(\mathbf{A}_i, \mathbf{A}_j)$ be a distance function to measure the relationship between the ith and jth attention matrices $\mathbf{A}_i$ and $\mathbf{A}_j$. Note that we reshape the 12 layers$\times$12 heads into 144 attention matrices because the inputs of $dist$ are two matrices. Intuitively, there should be three properties in a defined distance function. (i) $dist(\mathbf{A}_i, \mathbf{A}_i) = 0$; (ii) the distance function should be symmetric, i.e., $dist(\mathbf{A}_i, \mathbf{A}_j) = dist(\mathbf{A}_j, \mathbf{A}_i)$; (iii) If $\mathbf{A}_i$ is more similar to $\mathbf{A}_j$ than $\mathbf{A}_k$, then $dist(\mathbf{A}_i, \mathbf{A}_j) < dist(\mathbf{A}_i, \mathbf{A}_k)$. So for the following distances, we modify them according to these requirements and normalize them into range [0,1] for easy comparisons.

4.1 Two Levels of Distances

Token-Based Distance For one input sentence, we can extract 144 attention weight matrices. Each matrix is $\in [0,1]^{n\times n}$ where n is the number of tokens in the sentence. Then for each *token* in a sentence, we get 144 attention vectors ($\in [0,1]^n$). We can compute the pairwise distance of them and average on the n tokens to obtain a final scalar value for a pair of attention matrices.

Four token-based distance functions are generated from the following: cosine similarity (**cos**), Pearson correlation coefficient (**corr**), Jensen-Shannon distance (**JS**), and Bhattacharyya coefficient (**BC**). Note that for readability, please refer to Appendix A for detailed descriptions, implementations, and our modifications.

Sentence-Based Distance Unlike the token-based distances, here we directly measure the distance between two $n \times n$ attention matrices corresponding to the whole input *sentence*. We modified three measures: distance correlation (**dCor**) (Székely et al., 2007), Procrustes coefficient (**PC**) (Gower, 1971), and Canonical correlation coefficient (**CC**) (Hotelling, 1992). Please refer to Appendix A for more details.

In general, sentence-based measures care about covariance and/or linear/nonlinear dependency for the given two attention matrices while token-based ones only focus on two single attention vectors.

4.2 Visualization of Redundancy Matrices

We take CoLA with pre-trained BERT-base as an example to visualize the redundancy matrices based on the two sets of distance functions. We also conduct visualization for other GLUE tasks in Appendix B. Shown in Figure 2, each heatmap shows the normalized distances (averaged over the 1000 selected CoLA development data) using one distance function. Each entry reflects the distance between a pair of attention heads. We can see clear common redundancy patterns existing in the redundancy matrices based on token-based distances. Similar observations are shown for sentence-based distances.

5 Where: Redundancy in Layers

In this section, we discuss the redundancy patterns in details and show that similar redundancy patterns exist in redundancy matrices when using the four token-based distance functions. So do for the three sentence-based distance functions.

5.1 Redundancy Patterns

There are two observations in Figure 2. First, attention heads in the same layer and consecutive layers

[2] If the total number of development data is less than 1000, we select all data.

[3] https://github.com/huggingface/transformers

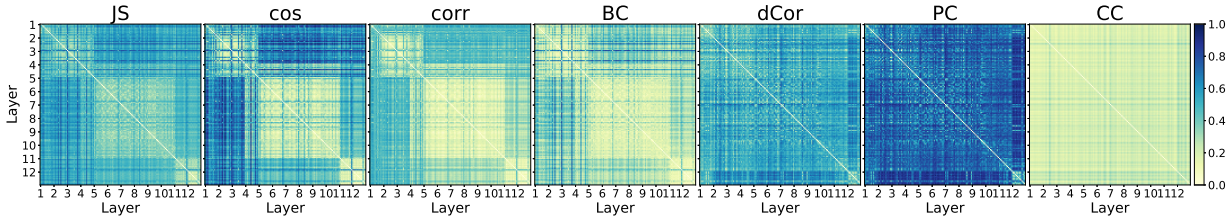


Figure 2: Redundancy matrices in the pre-trained BERT-base model for CoLA development data using various distance functions. The left four and right three are for token-based and sentence-based distances, respectively. Similar attention redundancy patterns (cluster with small distances) exist in each set of distance functions.

tend to be more redundant than heads in far-apart layers. Second, redundancy hierarchical cluster structure exists among layers. However, different cluster structures exist as well when comparing results of the two sets of distance measures. The left four sub-figures are for token-based distances. They cluster the first 4 layers, the middle 6 layers and the last 2 layers into three clusters and the last 8 layers into a bigger cluster. The last three sub-figures are for sentence-based measures. The cluster structures are not clearer than that of token-based measures. But we can still observe that the middle 10 layers have smaller distances. Heads in the same layer have closer distances, especially the first and last layer. From the view of attention redundancy, these observations can be explained by conclusions in Kovaleva et al. (2019); Clark et al. (2019) that heads within the same layer or nearby attention layers are extracting similar features (e.g., task-specific features).

5.2 Similar Patterns

Next, we show that similar redundancy patterns occur when the four token-based distances are employed. So do the three sentence-based distances. In Figure 3, we compute the correlation[4] between each pair of the redundancy matrices (with CoLA and pre-trained BERT-base) in Figure 2. High correlations are observed in redundancy matrices of each type of distances. Note that we also provide experimental results for other tasks with both pre-trained and fine-tuned BERT-base model in Appendix C.

As for the different cluster structures for token-based and sentence-based distance functions, the reason is related to the information captured by different distance functions. Sentence-based measures consider the covariance or nonlinear relationship between two sets of attention matrices. But token-based ones target on two single attention vectors. Also, different distance functions measure relationships in different ways. For example, The sentence-based dCor distance is based on nonlinear operations and 1 is obtained if and only if the two sets of input random variables are independent[5]. The token-based Pearson correlation (corr) is a measure for linear correlations.

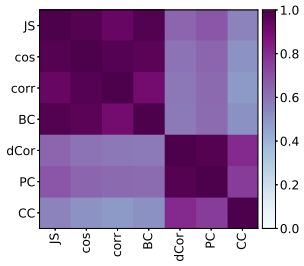


Figure 3: Correlation in two types of redundancy matrices (with pre-trained BERT-base for CoLA). There are high correlations among token-based and sentence-based distances, respectively.

In summary, by utilizing token-based and sentence-based distances to measure the attention redundancy, highly correlated similar redundancy patterns exist in each type, respectively.

6 When: Redundancy Is Phase-Independent

As a pre-trained language model, BERT is fine-tuned for downstream tasks. In this section, we check the redundancy patterns in both pre-trained and fine-tuned BERT-base model.

Figure 4 shows the attention redundancy matrices of fine-tuned BERT-base for CoLA using the two levels of distances (results for other tasks are shown in Appendix B). Very similar redundancy patterns to those in Figure 2 can be observed.

[4]Similar normalization and modification in Section 4.1 are conducted without being subtracted by 1.

[5]dCor distance = 1 is equivalent to that the original dCor correlation = 0 since we modify it with being subtracted by 1.

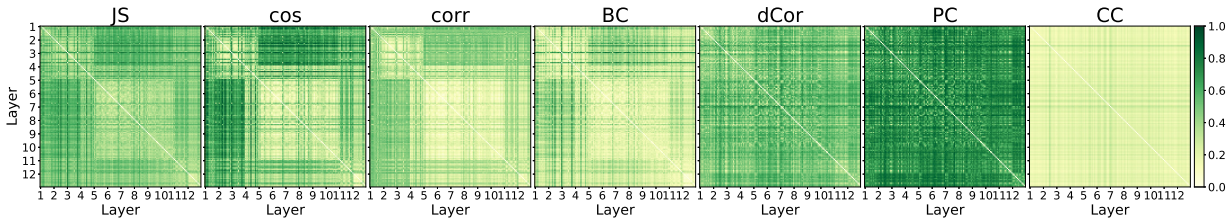


Figure 4: Redundancy matrices in fine-tuned BERT-base for CoLA. They are very similar to redundancy patterns in pre-trained BERT-base shown in Figure 2.

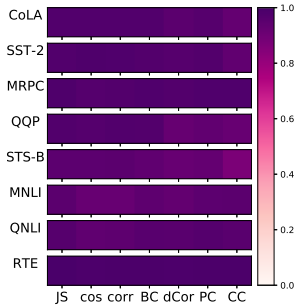


Figure 5: Correlation of redundancy matrices between pre-trained and fine-tuned BERT-base model for GLUE tasks. Highly correlated attention redundancy patterns occur in the two phases.

To quantify the similarity, we simply compute the absolute values of differences between two corresponding redundancy matrices in Figure 2 and Figure 4. The mean difference value is just 0.035. In addition, we compute correlations between pre-trained and fine-tuned BERT-base for each GLUE task under each distance function. As shown in Figure 5, high correlations are shown in pre-trained and fine-tuned phases for all GLUE tasks and all distance functions.

To summarize, attention redundancy patterns in pre-trained and fine-tuned phases are very similar and highly correlated.

7 Who: Redundancy Is Task-Agnostic

In this section, we analyze the attention redundancy from the task-oriented perspective. Specifically, for each distance function and each phase, we compute the correlation between redundancy matrices of each pair of tasks. The correlation results for pre-trained BERT-base model under different distances are shown in Figure 6 (results for fine-tuned BERT-base are presented in Appendix D). We find that the redundancy patterns across tasks are very similar with each other. Based on the task categories provided in Wang et al. (2018), we can also notice that relatively higher correlations occur in the two single-sentence tasks (CoLA and SST-2), the three similarity and paraphrase tasks (MRPC, QQP, and STS-B), and the three inference tasks (MNLI, QNLI, and RTE), respectively.

To further check the task-agnostic observation, we even randomly generate 1000 token sequences with various lengths (uniformly distributed) as input for the pre-trained BERT-base ("**RANDOM**" in Figure 6). We can observe high correlations among redundancy patterns of GLUE datasets and random inputs.

We conclude that attention redundancy patterns are task-agnostic. We think that it may be caused by the input formatting in BERT. In fine-tuning, the special token [CLS] and [SEP] are inserted in the beginning and at the end of one input sentence, respectively. If there are two sentences as input, they are also delimited by the special token [SEP]. It is found that [CLS] and [SEP] may dominate the attention distribution in some attention heads (Clark et al., 2019; Kovaleva et al., 2019) such that high similarities are observed through different tasks.

Based on this surprising discovery, we can apply this task-agnostic observation for pruning redundant attention heads. Without knowing any data of downstream task, we conduct a simple but effective clustering-based zero-shot pruning strategy based on attention redundancy in the following.

7.1 Case Study Application: Zero-Shot Attention Head Pruning

In this section, we introduce a simple zero-shot pruning strategy based on the task-agnostic cluster structure of attention redundancy matrices. Note that there are some complicated pruning strategies in the literature, such as (Tang et al., 2019; Jiao et al., 2019; Fan et al., 2019; Wang et al., 2019; McCarley, 2019). Most of them compress pre-trained models during or after fine-tuning to re-

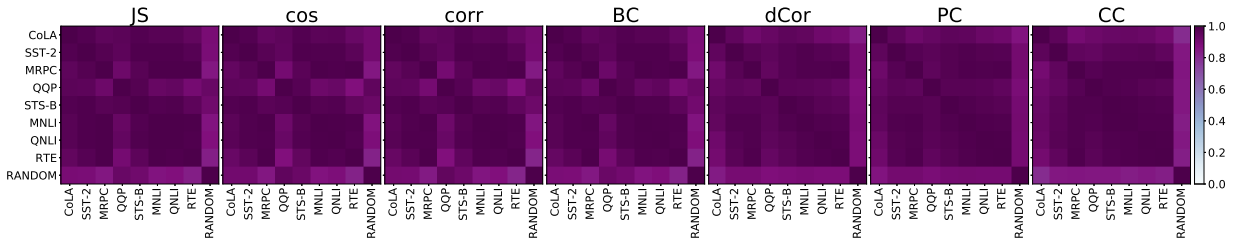


Figure 6: Correlations of redundancy matrices of task pairs (using pre-trained BERT-base). Redundancy patterns are task-agnostic. They are very similar across different tasks, even with random inputs.

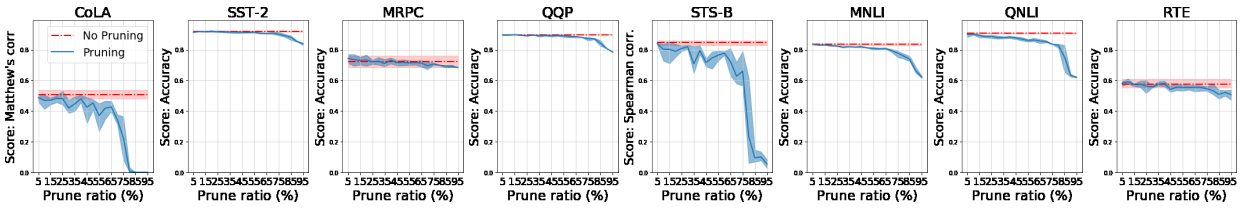


Figure 7: Performances of pruned BERT-base model for GLUE tasks. x-axis is the pruning ratio. Performances can be preserved after pruning even $75 - 85\%$ attention heads for most tasks.

duce the computational load in the inference phase. However, our proposed pruning strategy is before fine-tuning and zero-shot (i.e., without knowing any data in fine-tuning tasks).

Comparing and evaluating existing pruning works are beyond the scope of this paper. We leave the pruning topic as a future work. However, we believe that attention redundancy analyses in this paper benefit developing efficient pruning methods.

7.1.1 Pruning Method

The whole procedure is as the following. First, we feed randomly generated token sequences as input data in BERT-base model and obtain the redundancy matrices like Figure 2. Second, a clustering algorithm is applied on one redundancy matrix or the averaged redundancy matrices. If a pruning ratio p is given and the clustering method needs a given number of cluster, then we set the cluster number to be $\lceil 144 \times (1-p) \rceil$. Otherwise, a cluster-number-free clustering method is preferred. Third, a clustering goodness metric is used for each object (i.e., attention head) to obtain the cluster representative head which is kept during pruning. We can simply prune trainable parameters corresponding to other heads and relevant parameters in subsequent feed-forward layers, then fine-tune as usual for downstream tasks.

7.1.2 Pruning Performance

In this section, we applied the well-known spectral clustering[6] and Silhouettee Score[7] as the clustering goodness metric to obtain cluster representative heads. We prune attention heads using the same obtained pruning strategy (since the pruning is zero-shot and task-agnostic) and fine-tune the pruned BERT-base model for each GLUE task with suggested hyper-parameters in Section 3. The fine-tuned performances on development sets are shown in Figure 7. Each sub-figure corresponds to one task. x-axis is the pruning ratio between 5% and 95% with an interval of 5%. We conduct 10 trials for each fine-tuning task. Red dashed line reflects the average performance without pruning[8]. Blue line plots averaged performances under different pruning ratios. Boundaries of shadow areas cover the best and worst results of the 10 trials.

In Figure 7, not surprisingly, performances drop as the pruning ratio increases. However, in 4 (SST-2, MRPC, QQP and RTE) out of these 8 tasks , we could use a pruning ratio as big as 85%, without significant performance loss (< 5%) against an unpruned model. In MNLI and QNLI, the performance loss is bigger. But still, the pruning ratio

[6] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html

[7] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

[8] We obtain comparable or even better performances as the reported scores (https://github.com/huggingface/transformers/tree/master/examples/text-classification) averaged on 10 trials.

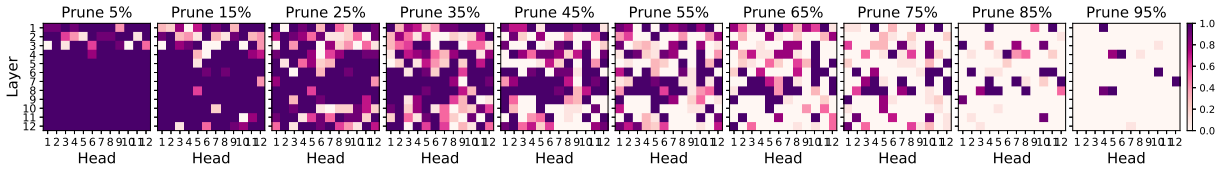


Figure 8: Chances of a head being pruned at various pruning ratios when using our redundancy clustering based zero-shot pruning strategy (BERT-base model with random inputs). Results are averaged over 10 clustering trials. In each heatmap, the title shows the pruning ratio (larger pruning ratios reflect that more heads are pruned). The lighter the entry color is, the more often the head gets pruned. We observe that heads in earlier and deeper layers are pruned with high chances under larger pruning ratios. Please refer to Section 7.1.3 for more details.

can be as big as 75% to guarantee a small performance loss (< 5%). The only two outliers are CoLA and STS-B. We think their smaller validation sets ($\leq$ 1000) may result in fluctuations and prevent us from making a strong conclusion, and a further study is deferred to future work.

7.1.3 Pruning Heads Visualization

In Figure 8, we visualize some pruning results averaged over 10 clustering trials. Lighter colors represent that one head is pruned more often. When pruning ratios are small (e.g., <15%), heads in the first four layers are pruned with higher chances than those in further layers. When pruning ratio increases (25%$\sim$75%), heads in earlier and deeper layers are pruned more often. However, some heads are always kept, for example, Head-2,6,10,12 in Layer-11 and Head-1,3,12 in Layer-12. In extreme cases (85% and 95% pruning ratios), pruning these heads hurts fine-tuning performances (Figure 7). Interestingly, along all pruning ratios (including cases of 85% and 95%), some heads in the middle layers are kept with high chances.

We observe that heads in the first four layers are always very likely to be pruned. This verifies the cluster structure in redundancy matrices (e.g., Figure 2). This may due to the fact that heads in earlier layers capture more superficial linguistic features (Kovaleva et al., 2019) which might be less informative in fine-tuning than other heads. As pruning ratio increases, heads in deeper layers are also more likely to be pruned, than those in the middle layers. We conjecture that after the middle layers, the contextualized embeddings are already very "strong" for down-stream tasks. Therefore the deeper layers do not require too many heads (though a few are left) to handle the task.

To summarize the case study of pruning, we emphasize that the proposed simple but robust redundancy clustering based pruning method is task-agnostic and zero-shot. Compared to other pruning methods, (i) it requires no data from down-stream tasks; (ii) one pruning strategy obtained from the pre-trained model can robustly prune less informative heads and preserve comparable fine-tuning performances for all GLUE downstream tasks. The powerful strength of the simple pruning strategy results from the phase-independent and task-independent attention redundancy patterns existing in BERT-base model. As a future work, we would check if similar attention redundancy patterns exist in other multi-layer multi-head self-attention based models and develop corresponding pruning mechanisms.

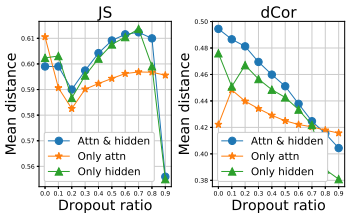


Figure 9: The effects of dropout ratio on the attention redundancy. "N"-shape is shown on the left (token-based JS distance). Almost monotonic effects exist on the right (sentence-based dCor distance): higher dropout results in more redundancy. Redundancy is more sensitive to *hidden-dropout-ratio* than *attention-head-dropout-ratio*.

8 "Why": Effects of Pre-Training Dropouts

It is hard to answer the fundamental question why attention redundancy exists. However, in this section, we examine one factor that may be the cause.

Dropout in the pre-training phase is suspected to be one reason resulting in redundant attentions (Clark et al., 2019). But Clark et al. (2019) didn't evaluate that. In this section, we investigate effects of various dropout ratios in the pre-training

process on the attention redundancy. In BERT pre-training, there are two dropout ratios, *attention-head-dropout-ratio* and *hidden-dropout-ratio*. The former randomly deactivates a ratio of trainable weights in the attention head (key, query, and value transformation matrices). The latter deactivates some weights in the linear transformation matrices which integrate all attention heads after each attention layer. Their default values are 0.1.

We manually set various values (from 0.0 to 0.9) for those two dropout ratios and train a BERT masked language model on Wiki103 training dataset[9] from scratch. We train 100 epochs or until training loss converges. Then we randomly select 1000 sentences from the test dataset as objects. For each sentence, we calculate the mean value of attention redundancy matrix using JS and dCor distance. The final reported values are averaged over the 1000 sentences.

Results of three settings are shown in Figure 9. "Only attn" means that we only change the *attention-head-dropout-ratio* and keep the *hidden-dropout-ratio* as the default value 0.1. "Only hidden" is changing *hidden-dropout-ratio* and keeping *attention-head-dropout-ratio* as default. "Attn & hidden" modifies both dropout ratios. In both figures, smaller distance values reflect heavier attention redundancy.

We can see that the token-based measure JS and sentence-based measure dCor show reversed trends in the middle range ([0,2 0.7]). For JS on all three settings, when dropout ratio increases (i.e., more inactive weights updating in pre-training) the distances first drop down to the lowest values (when dropout ratio is 0.2) and increase and then drop again heavily. On the other hand, for dCor, the distance decreases (heavier attention redundancy) along the increased dropout ratio.

We conclude that dropout ratio does not play a simple effect on the attention redundancy. The "N"-shape effects shown in the left figure (token-based distance JS) is demonstrated from the view of attention vectors' similarity. On the other hand, the sentence-based distance (dCor) figure shows that higher dropout ratios result in more redundancy[10].

In both figures, we observe that slopes of "Attn & hidden" and "Only hidden" are steeper than that of "Only attn". This means that dropout in the hidden transformations affects the attention redundancy more than the attention dropout.

There is no doubt that more factors must affect the attention redundancy. We leave the "why" in future study.

9 Conclusion

Using BERT-base model as an example, we comprehensively investigated the attention redundancy in multi-layer multi-head self-attention based language models. The redundancy was measured by distance functions at token level and sentence level. At both levels, we found that many heads are not distinct from each other, and clear clustering effects were observed. We discovered that the attention redundancy is phase-independent and task-agnostic. Specifically, compared to a pre-trained model, the redundancy patterns do not change much after fine-tuning on multiple downstream tasks. We also shown complex influences on redundancy of dropout ratios in hidden transformations and self-attention. Based on these discoveries, we design a zero-shot strategy to prune attention heads. Compared to existing methods, the zero-shot pruning is simple and robust (task-agnostic). In the near future, we are interested in experimenting this method over more self-attention based pre-trained language models and more downstream tasks.

[9] https://blog.einstein.ai/the-wikitext-long-term-dependency-language-modeling-dataset/

[10] Note that we use 1 - original distance correlation for consistency among distance functions. Original distance correlation=0 means that two sets of random variables are independent.

References

Anil Bhattacharyya. 1946. On a measure of divergence between two multinomial populations. *Sankhyā: the indian journal of statistics*, pages 401–406.

Xingyu Cai, Jiaji Huang, Yuchen Bian, and Kenneth Church. 2021. Isotropy in the contextual embedding space: Clusters and manifolds. In *ICLR*.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*.

Allyson Ettinger. 2020. What bert is not: Lessons from a new suite of psycholinguistic diagnostics for language models. *Transactions of the Association for Computational Linguistics*, 8:34–48.

Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing transformer depth on demand with structured dropout. In *ICLR*.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

JC Gower. 1971. Statistical methods of comparing different multivariate analyses of the same data. *Mathematics in the archaeological and historical sciences*, pages 138–149.

Harold Hotelling. 1992. Relations between two sets of variates. In *Breakthroughs in statistics*, pages 162–190. Springer.

Sarthak Jain and Byron C Wallace. 2019. Attention is not explanation. In *NAACL*.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does bert learn about the structure of language? In *ACL*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Julie Josse and Susan Holmes. 2016. Measuring multivariate association and beyond. *Statistics surveys*, 10:132.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of bert. *arXiv preprint arXiv:1908.08593*.

Pierre Legendre and MARIE-JOSÉE FORTIN. 2010. Comparison of the mantel test and alternative approaches for detecting complex multivariate relationships in the spatial analysis of genetic data. *Molecular ecology resources*, 10(5):831–844.

Yongjie Lin, Yi Chern Tan, and Robert Frank. 2019. Open sesame: Getting inside bert's linguistic knowledge. *arXiv preprint arXiv:1906.01698*.

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. *arXiv preprint arXiv:1907.11692*.

J Scott McCarley. 2019. Pruning a bert-based question answering model. *arXiv preprint arXiv:1910.06360*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *NeurIPS*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A primer in bertology: What we know about how bert works. *arXiv preprint arXiv:2002.12327*.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2019. Ernie 2.0: A continual pre-training framework for language understanding. *arXiv preprint arXiv:1907.12412*.

Gábor J Székely, Maria L Rizzo, Nail K Bakirov, et al. 2007. Measuring and testing dependence by correlation of distances. *The annals of statistics*, 35(6):2769–2794.

Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136*.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel Bowman, Dipanjan Das, et al. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. In *ICLR*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. *arXiv preprint arXiv:1906.04284*.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *ACL*.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do nlp models know numbers? probing numeracy in embeddings. In *EMNLP*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2019. Structured pruning of large language models. *arXiv preprint arXiv:1910.04732*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

This is the appendix for NAACL-HLT 2021 paper: Yuchen Bian, Jiaji Huang, Xingyu Cai, Jiahong Yuan, and Kenneth Church. On Attention Redundancy: A Comprehensive Study.

A Distance Functions

We provide detailed descriptions of distance functions and our modifications in this section. Most of them exist in python scipy and sklearn packages. For others, we also provide implementation references or implement by ourselves. But more mathematical details are beyond this paper's scope. Please refer to original papers or (Josse and Holmes, 2016) for discussions.

A.1 Token-Based Distances

For an input sentence, each token corresponds to 144 attention vectors.

Let $\mathbf{p}, \mathbf{q}$ be two attention vectors or attention distributions (since the sum of each attention vector is 1). Four modified distance functions used in this paper are:

- *cosine similarity (**cos**):* Since $cos(\mathbf{p}, \mathbf{q})$ for a pair of distribution vectors is bounded in $[0, 1]$, we modify it with $1 - cos(\mathbf{p}, \mathbf{q})$ to keep the distance properties in Section 4.

- *Pearson correlation coefficient (**corr**):* We normalize as $(1 - corr(\mathbf{p}, \mathbf{q}))/2$ due to its range $[-1, 1]$.

- *Jensen-Shannon distance (**JS**):* Its range is [0,1] and it's consistent with the distance properties. It is also used in the literature (Clark et al., 2019; Jain and Wallace, 2019) to measure the distance of two attention distributions.

- *Bhattacharyya coefficient (**BC**) (Bhattacharyya, 1946):* It measures the amount of overlap between two statistical populations. It can be used to determine the relative closeness of the two attention vectors. $BC(\mathbf{p}, \mathbf{q}) = \sum_x \mathbf{p}(x)\mathbf{q}(x)$. Its range is $[0, 1]$, and we modify it by $1 - BC(\mathbf{p}, \mathbf{q})$.

A.2 Sentence-Based Distances

For an input sentence, let $\mathbf{A}_i$ and $\mathbf{A}_j$ be arbitrary two attention matrices among the 144 attention matrices extracted from the BERT-base model.

- *Distance correlation*[11] *(**dCor**) (Székely et al., 2007):* It is introduced to address the deficiency of Pearson's correlation which is sensitive to a linear relationship between two variables. It's widely used in statistical community. It's based on nonlinear operation and the range is $[0, 1]$ where 0 is got when two sets of random variables are independent. We modify it as 1-$dCor(\mathbf{A}_i, \mathbf{A}_j)$.

- *Procrustes coefficient (**PC**) (Gower, 1971) :* It can measure the closeness of two data matrices. It's also known as Lingoes and Schönemann (RLS) coefficient (Legendre and FORTIN, 2010). It varies from 0 to 1 and can be used as a distance measure.

- *Canonical correlation coefficient (**CC**) (Hotelling, 1992):* It's a famous method to study the relationship between two sets of variables. It's defined as the trace of a matrix combining the covariance of two input data matrices. We modify it as 1-$CC(\mathbf{A}_i, \mathbf{A}_j)$.

B Attention Redundancy Matrices

In this section, we provide attention redundancy matrices visualization results for GLUE datasets and randomly generated token sequences (Figure 10 to Figure 18).

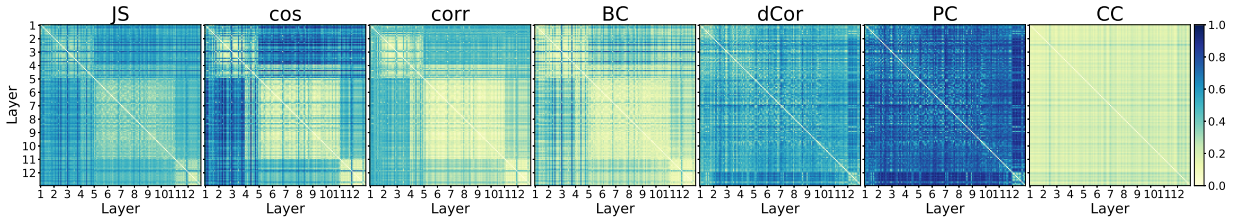
C Consistency of Redundancy Patterns in GLUE Tasks

In this section, we provide more consistency results of attention redundancy patterns measured based on token-based and sentence-based distances for GLUE tasks in Figure 19 including both pre-trained and fine-tuned BERT-base model.

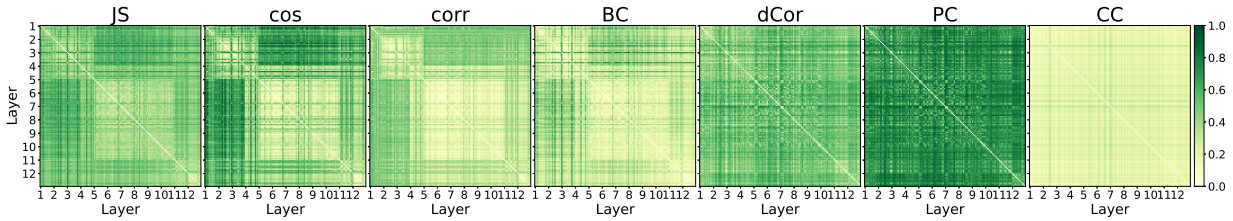
D Cross-Task Correlations of Redundancy Patterns in GLUE Tasks

In this section, we show the cross-task correlation results of attention redundancy patterns measured based on different distances for GLUE tasks and random inputs in fine-tuned BERT-base model in Figure 20.

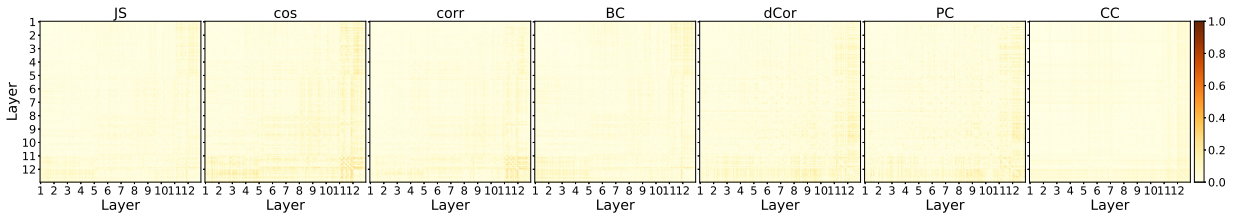
[11] https://github.com/vnmabus/dcor



(a) Redundancy matrices in pre-trained BERT-base for CoLA

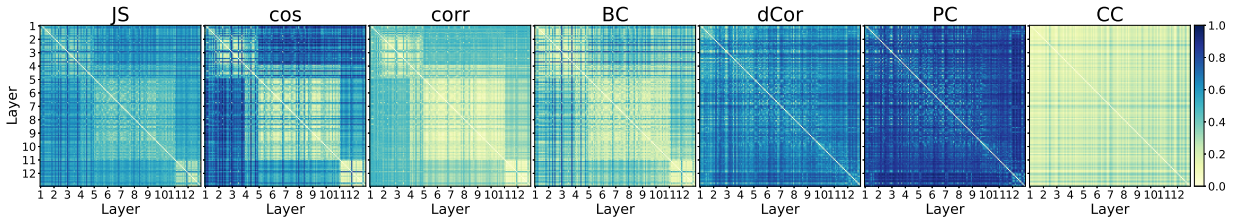


(b) Redundancy matrices in fine-tuned BERT-base for CoLA

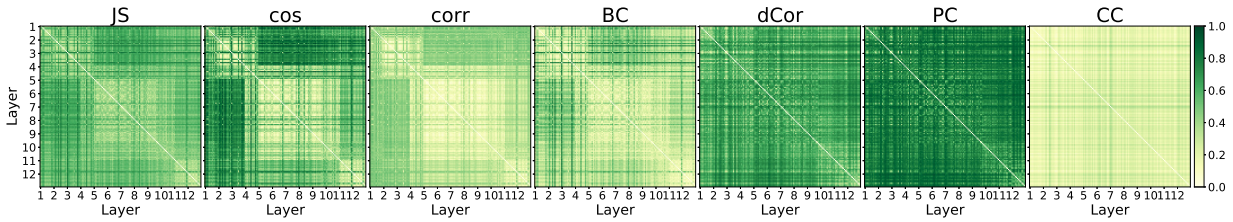


(c) Difference of redundancy matrices between pre-trained and fine-tuned BERT-base for CoLA

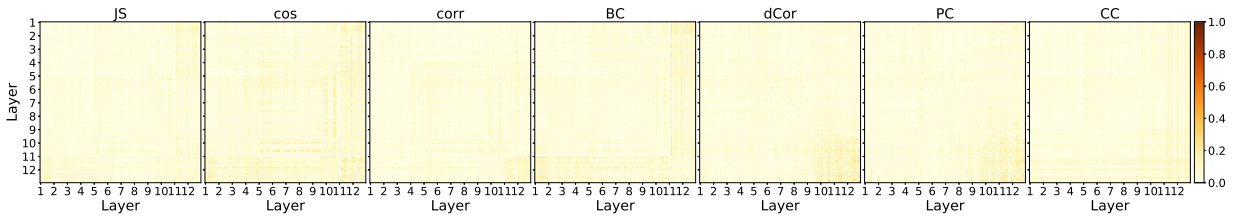
Figure 10: Redundancy matrices in BERT-base for CoLA



(a) Redundancy matrices in pre-trained BERT-base for SST-2

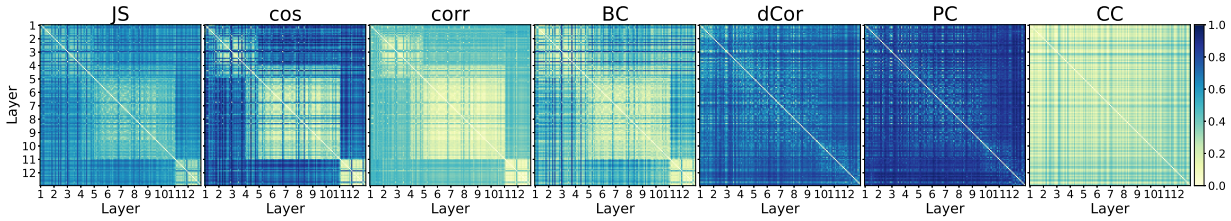


(b) Redundancy matrices in fine-tuned BERT-base for SST-2

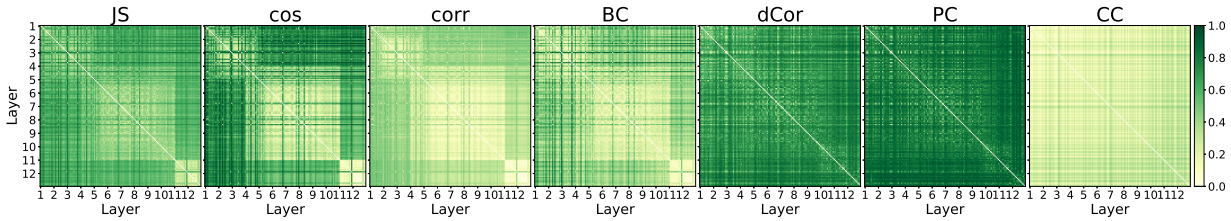


(c) Difference of redundancy matrices between pre-trained and fine-tuned BERT-base for SST-2

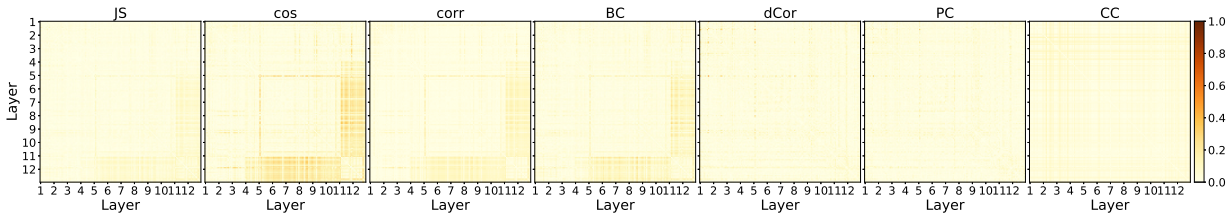
Figure 11: Redundancy matrices in BERT-base for SST-2



(a) Redundancy matrices in pre-trained BERT-base for MRPC

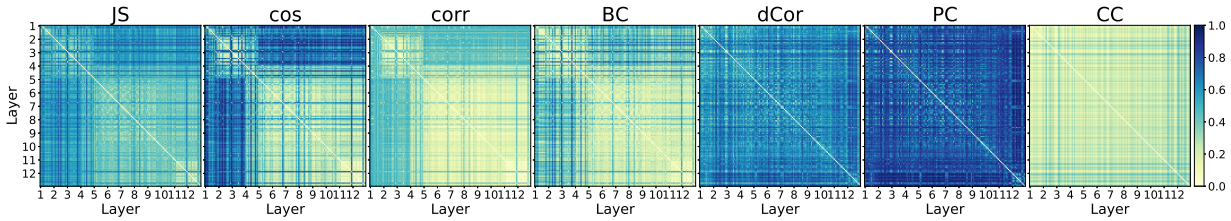


(b) Redundancy matrices in fine-tuned BERT-base for MRPC

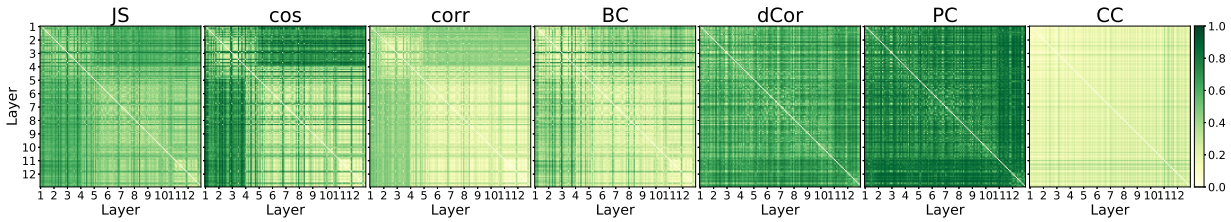


(c) Difference of redundancy matrices between pre-trained and fine-tuned BERT-base for MRPC

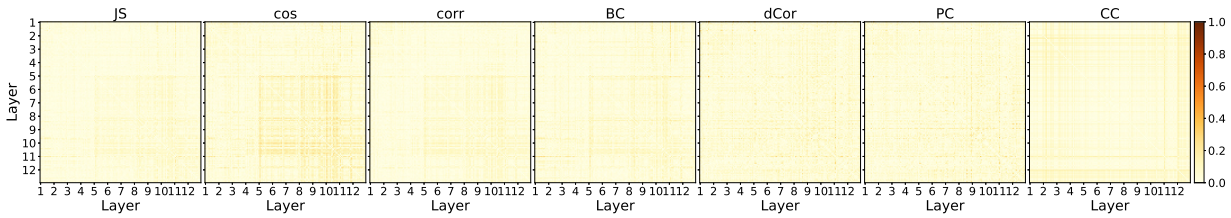
Figure 12: Redundancy matrices in BERT-base for MRPC



(a) Redundancy matrices in pre-trained BERT-base for QQP

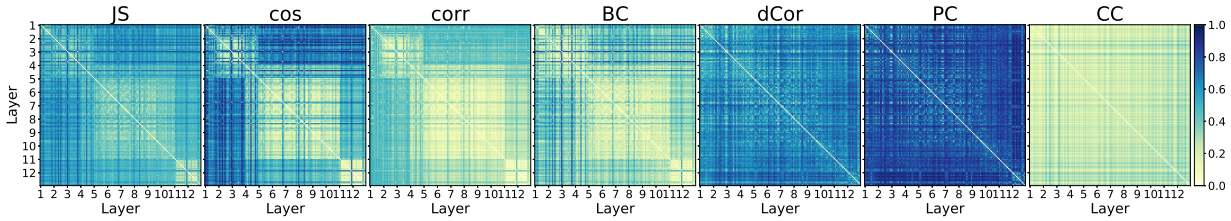


(b) Redundancy matrices in fine-tuned BERT-base for QQP

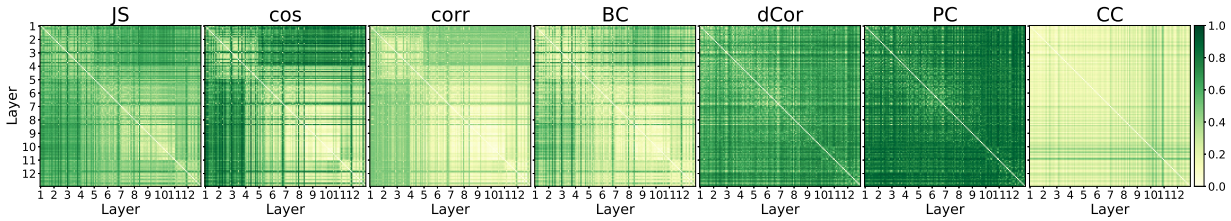


(c) Difference of redundancy matrices between pre-trained and fine-tuned BERT-base for QQP

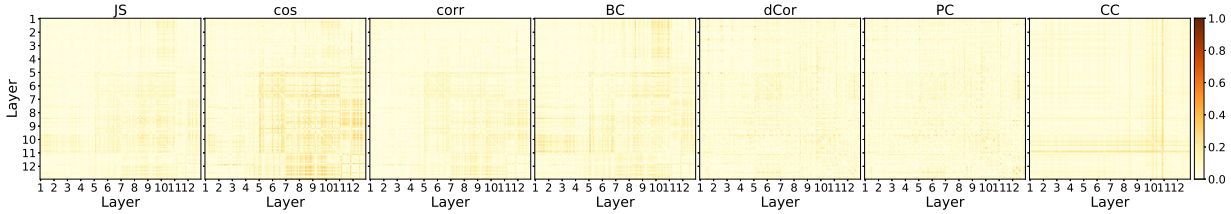
Figure 13: Redundancy matrices in BERT-base for QQP



(a) Redundancy matrices in pre-trained BERT-base for STS-B

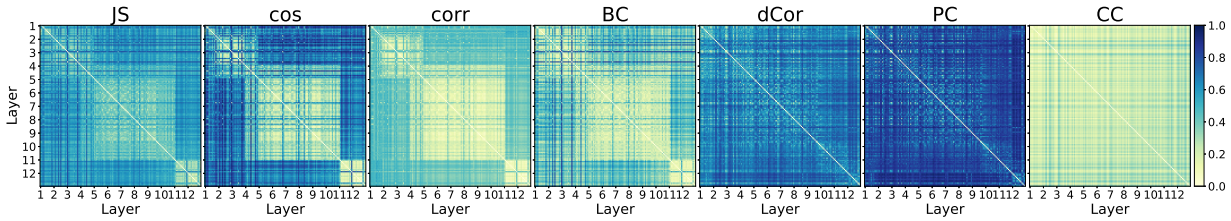


(b) Redundancy matrices in fine-tuned BERT-base for STS-B

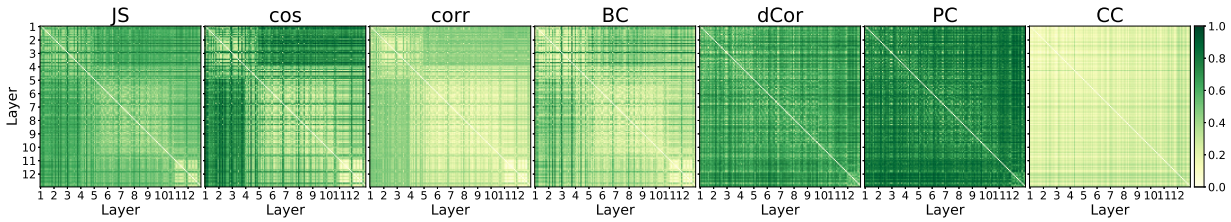


(c) Difference of redundancy matrices between pre-trained and fine-tuned BERT-base for STS-B

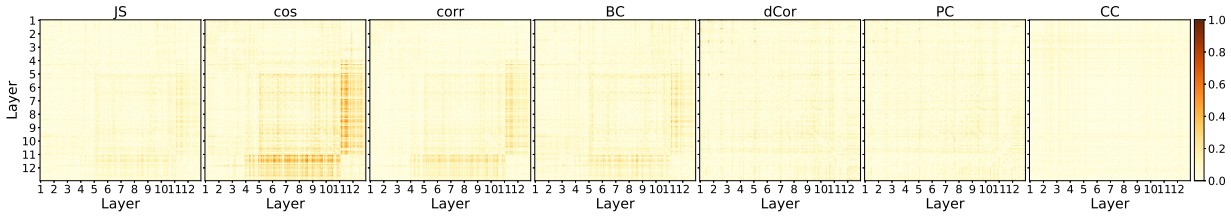
Figure 14: Redundancy matrices in BERT-base for STS-B



(a) Redundancy matrices in pre-trained BERT-base for MNLI

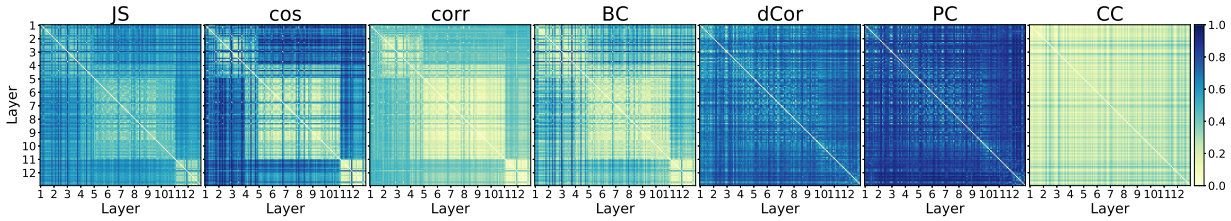


(b) Redundancy matrices in fine-tuned BERT-base for MNLI

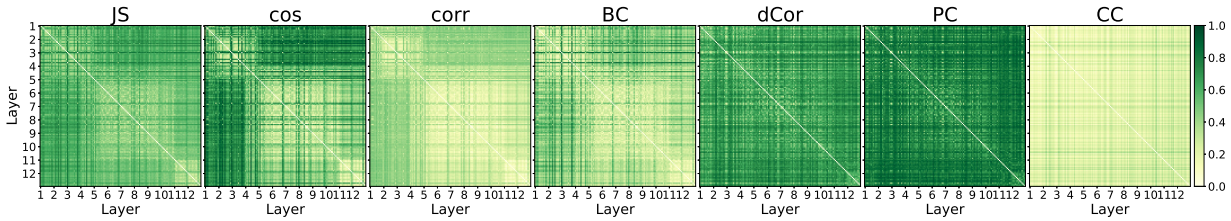


(c) Difference of redundancy matrices between pre-trained and fine-tuned BERT-base for MNLI

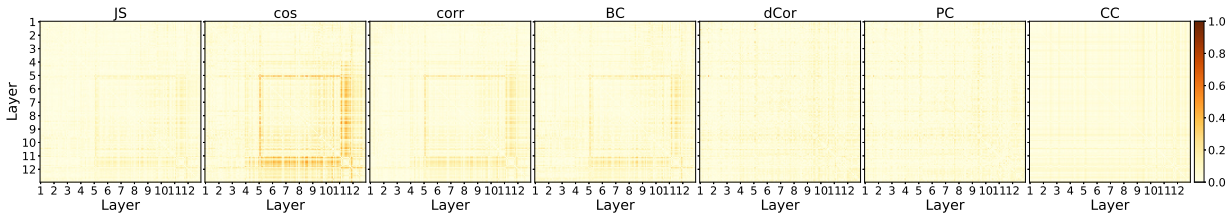
Figure 15: Redundancy matrices in BERT-base for MNLI



(a) Redundancy matrices in pre-trained BERT-base for QNLI

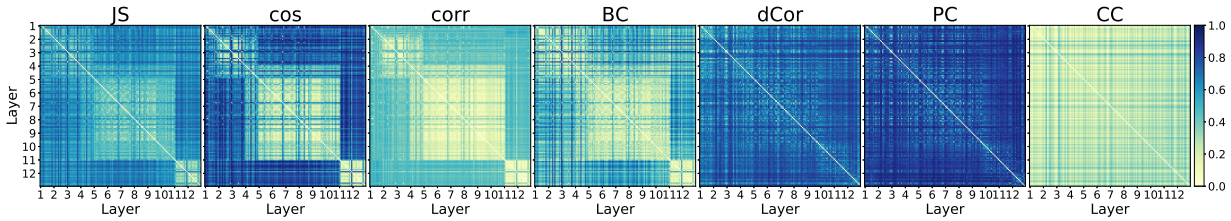


(b) Redundancy matrices in fine-tuned BERT-base for QNLI

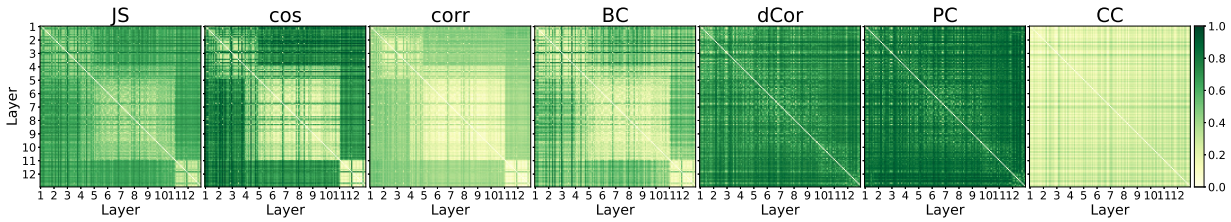


(c) Difference of redundancy matrices between pre-trained and fine-tuned BERT-base for QNLI

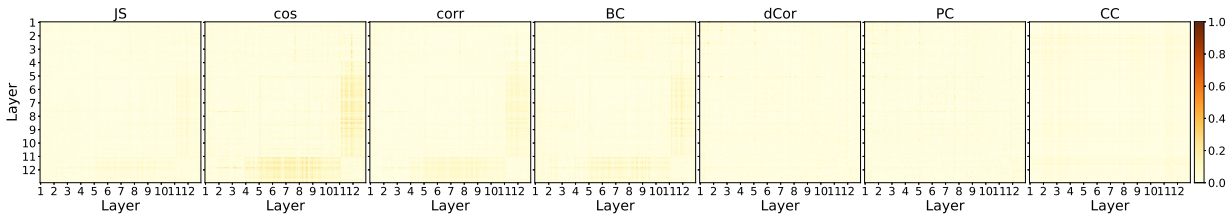
Figure 16: Redundancy matrices in BERT-base for QNLI



(a) Redundancy matrices in pre-trained BERT-base for RTE



(b) Redundancy matrices in fine-tuned BERT-base for RTE



(c) Difference of redundancy matrices between pre-trained and fine-tuned BERT-base for RTE

Figure 17: Redundancy matrices in BERT-base for RTE

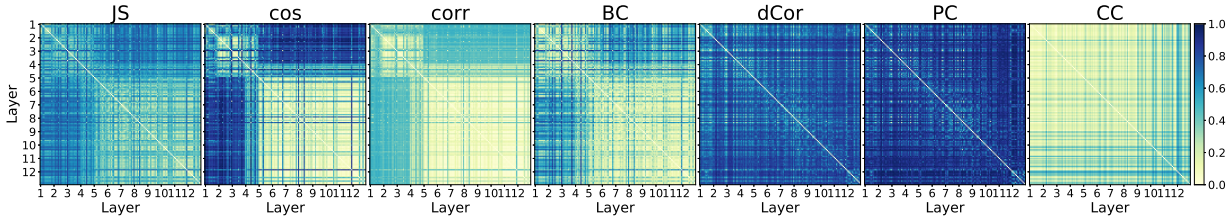
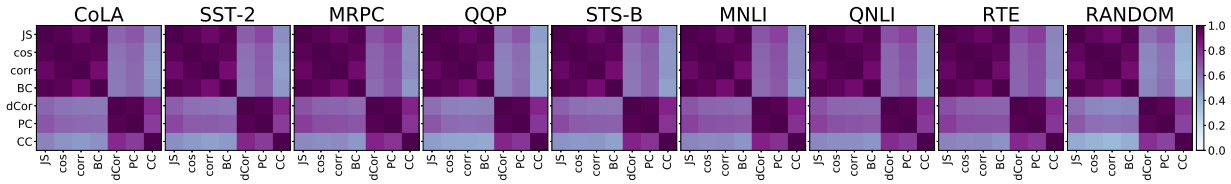
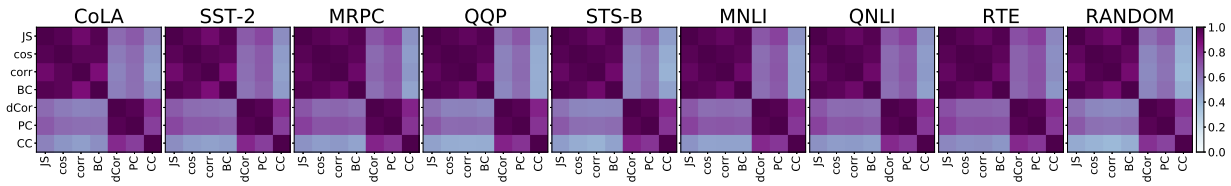


Figure 18: Redundancy matrices in pre-trained BERT-base for randomly generated token sequences



(a) Pre-trained BERT-base model



(b) Fine-tuned BERT-base model

Figure 19: Correlation of redundancy matrices in BERT-base for GLUE tasks and random inputs

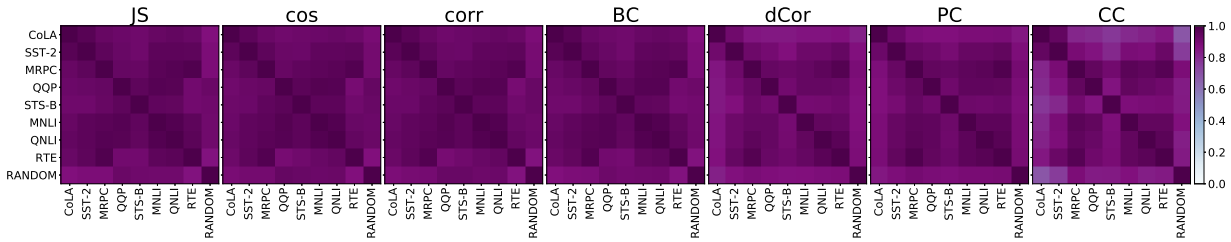


Figure 20: Correlations of redundancy matrices between task pairs (with fine-tuned BERT-base)