

Parameter Efficient Fine-tuning for Multi-task Learning

Stanford CS224N Default Project

Chih-Ying Liu

Department of Electrical Engineering
Stanford University
ying1029@stanford.edu

Jeffery Shen

Department of Computer Science
Stanford University
jshen7@stanford.edu

Abstract

This project aims to implement, utilize, and improve upon the BERT model to perform sentiment analysis and other downstream tasks. In the first part of our project, we fully implement the original BERT model and test it on sentiment analysis – in the second part, we fine-tune and extend the model for optimal performance on paraphrase detection and semantic textual similarity. To do so, we implement Projected Attention Layers (PALs) Stickland and Murray (2019), adapters Houshy et al. (2019), and prefix tuning Li and Liang (2021) to achieve optimal performance over multi-tasks while being efficient. We also experiment with changes to the BERT model architecture by implementing Sentence-BERT Reimers and Gurevych (2019) and modifying the downstream classifier head architecture.

We experiment with different model architectures, adaptation modules, samplers, adding additional training data, and hyper-parameter configurations. We find Sentence-BERT learns more semantically meaningful sentence embeddings and has better performance on the paraphrase and similarity tasks. PAL, prefix, and adapter improve average performance by about 8% when pretraining, and have comparable performance to full fine-tuning with less than 10% of trainable parameters.

1 Key Information to include

- Mentor: Tathagat Verma
- External Collaborators: N/A
- Sharing project: N/A

2 Introduction

In this publication, we present an implementation of minBERT, a baseline BERT model for use in single-task learning. We then further expand on the model with the goal of optimizing for multi-task learning without having to finetune different models for individual tasks.

At release time, BERT was able to obtain new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (a 7.7% point absolute improvement) (Devlin et al., 2018). However, there are still areas to improve upon. One point of interest is parameter efficiency – BERT’s state-of-the-art results used transfer from unsupervised pre-training with BERT, with a separate BERT model fine-tuned for each individual task. Our goal is to introduce a BERT model that can build robust embeddings that perform well across a large range of different tasks, without having to finetune individual models for individual tasks. We aim to improve upon the minBERT model’s scores across the provided three tasks.

To do so, we will explore two main pathways of optimization: changes to the model architecture itself, and adding in additional adaptation modules. In our changes to the model architecture, we will implement Sentence-Bert by Reimers and Gurevych (2019), which should result in greatly increased performance on the STS task. We will also investigate modifications to the downstream classifier head architecture, modifying aspects such as layer type, number of layers, and how we feed the inputs through BERT. In addition to these changes, we will also implement additional adaptation modules such as Projected Attention Layers (PALs) Stickland and Murray (2019), prefix tuning Li and Liang (2021), and adapter Houlsby et al. (2019). We seek to investigate how these adaptation modules can achieve optimal performance while being parameter efficient, and also will experiment with different sizes (configurations) of these modules.

3 Related Work

BERT is a transformer-based model that generates contextual word representations. BERT uses a cross-encoder: Two sentences (with a SEP token in-between) are passed to the transformer network and the target value is predicted. BERT, when first released, set new state-of-the-art results for various NLP tasks, including question answering (QA), sentence sentiment classification, and sentence-pair regression, such as semantic similarity (STS).

However, this setup is unsuitable for various pair regression tasks (such as STS) due to the amount of possible combinations and $O(n^2)$ number of inference computations required. Instead, to address these tasks, mapping each sentence to a vector space such that semantically similar sentences are close is usually a better approach – one common approach is to use the output of the [CLS] token. But, even these approaches result in sub-optimal performance, often even worse than just averaging GLoVe embeddings. To alleviate this issue, Sentence-BERT adds a pooling operation to the output of BERT to derive a fixed sized sentence embedding, computing the mean of all output vectors. (Reimers and Gurevych, 2019)

Prior state-of-the-art results across multiple natural language understanding (NLU) tasks have previously used transfer from a single large task: unsupervised pre-training with BERT, where a separate BERT model was fine-tuned for each downstream task. However, this requires a different model to be trained and stored for each individual task. With the goal of multi-task learning, there is a need to be parameter efficient. Taking this into account, Projected Attention Layers (PALs), a kind of adaptation module, allow for a multi-task approach that shares a single BERT model backbone with a small number of additional task-specific parameters. This allows for comparable performance as compared to separately fine-tuned models, using significantly less parameters. (Stickland and Murray, 2019)

For a more general perspective of transfer learning, in which we aim to transfer the knowledge of pre-trained model to various downstream tasks, it is parameter inefficient to fine-tune each task separately. As an alternative, adapter and prefix-tuning add only a few trainable parameters and keep the backbone model fixed. They achieve comparable performance to fine-tuning with less than 3.6% (adapter) and 0.1% (prefix tuning) trainable parameters. Furthermore, since the backbone model is fixed, new tasks can be added without revisiting the old ones, and therefore also eliminates negative task interference in multi-task settings. (Li and Liang, 2021) (Houlsby et al., 2019)

4 Approach

With the goal of improving upon the default BERT base model to achieve optimal multitask performance, we explore two main pathways of optimization: changes to the model architecture itself, and adding in additional adaptation modules.

Model Architecture We will experiment with changing the downstream classifier head architecture. First, the default classifier head only consists of a single linear layer – to improve performance, we will add an additional linear layer with an activation function in between. For example, for the sentiment classification task, our classifier head is:

$$\text{LinLayer1}(\text{hiddenSize} \rightarrow \text{hiddenSize}) \rightarrow \text{ReLU}() \rightarrow \text{LinLayer2}(\text{hiddenSize} \rightarrow \text{numLabels})$$

We also seek to implement cosine similarity for improvement on the Semantic Textual Similarity (SST) task. Instead of concatenating the 2 pooled outputs from BERT and passing them through a

linear layer, we will get the cosine similarity of the two outputs and then map the similarity (in the range $[-1, 1]$) to $[0, 5]$ (the range of outputs for SST) using the equation $(\cos\text{Similarity} + 1) * \frac{5}{2}$.

Additionally, we will also implement Sentence-BERT as described by Reimers and Gurevych (2019). Sentence-BERT feeds sentence pair inputs separately through the BERT model, and adds a mean pooling operation across sequence outputs to derive the sentence embedding. Sentence-BERT uses different model designs for different objective functions. For the semantic similarity task, we calculate the cosine similarity between sentence embeddings and use a regression objective. For the paraphrase classification task, we concatenate two sentence embeddings u and v with the absolute value of their element wise difference $|u - v|$, and feed these three vectors into our classifier head. Figure 1 shows our model architecture.

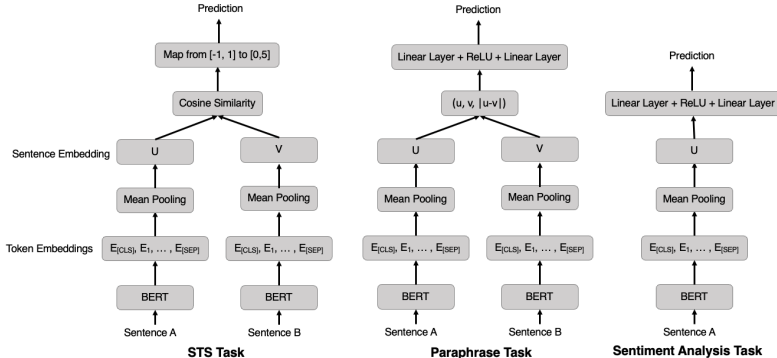


Figure 1: Model architecture for sentiment, paraphrase, and similarity tasks. We follow Sentence-BERT architecture for paraphrase and similarity tasks.

Adaptation Modules In our second exploration pathway, we seek to implement and add additional adaptation modules to the base BERT model.

The first method we will implement to extend the base model are PAL layers, with reference to Stickland and Murray (2019). PALs involve a task-specific function:

$$TS(h) = V^D g(V^E h)$$

where V^D and V^E are some projection layer shared across layers, and $TS(\cdot)$ is self attention function. This task-specific function is added parallelly to BERT layers, from the input of BERT layer to the last layer norm layer.

The second method we will implement is prefix-tuning from Li and Liang (2021). Prefix-tuning prepends some trainable parameters to the input of transformer layers. Since prefix tuning is sensitive to initialization, and random initialization leads to low performance and high variance, we initialize the prefix with hidden states calculated by the first batch of SST data. By doing so, we ensure that the prefix is more similar to meaningful words in the embedding space.

The third method we will implement is adapter from Houlsby et al. (2019). Adapter contains a linear down projection function, an activation function, a up projection function, and a skip connection from input to output. We add the adapter module twice to each BERT layer: once after the feed forward layer following multi-head attention and once after the two feed-forward layers. For adapter tuning, we learn task specific adapter modules, layer normalization parameter, and downstream heads. Figure 2 shows the architecture for these three adaptation modules.

Additional Methods Finally, we also seek to improve upon the default round-robin sampling method that the current base model uses when choosing which task it selects a datapoint to train from. We will implement the training scheduling method as described by Stickland and Murray (2019) for our training pipeline – proportional, square root, and annealed sampling. Instead of the traditional "round-robin" sampling of training examples (which can lead to overfitting on smaller datasets and vice versa for larger datasets), we will select examples from a task i with a probability:

$$p_i \propto N_i^\alpha$$

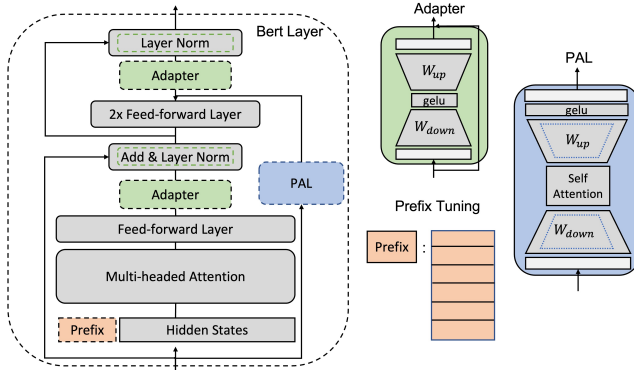


Figure 2: Illustration of BERT layer with PAL, prefix, and adapter.

When $\alpha = 1$, the above equation represents proportional sampling. If $\alpha < 1$, we reduce the disparity between choosing tasks. When $\alpha = .5$, this is square root sampling. When α changes with each epoch e (and the total number of epochs is E), this is annealed sampling:

$$\alpha = 1 - 0.8 \frac{e - 1}{E - 1}$$

5 Experiments

Data The default datasets given to us include the Stanford Sentiment Treebank (SST) dataset, the CFIMDB dataset, the Quora (QQP) dataset, and the SemEval STS Benchmark dataset. These are described in Socher et al. (2013), Agirre et al. (2013), and the project proposal. However, we notice that our collection of datasets is heavily skewed towards the Paraphrase detection task. To alleviate concerns of poor performance on the SA and STS tasks due to insufficient data, we explored adding additional data to provide more context. In the interest of space, detailed information about the additional datasets and their formatting is in A.1.

Name	Task?	Size (Total)	Size (Train)
Stanford Sentiment Treebank (SST)	SA	11,855	8,544
CFIMDB	SA	2,438	1,705
Quora (QQP)	Paraphrase	202,151	141,506
SemEval STS	STS	8,628	6,040
SemEval SICK 2014	STS	10,000	4,500
Amazon Kindle Reviews	SA	982,619	<i>variable</i>
Rotten Tomatos	SA	634,251	<i>variable</i>

Table 1: The above table describes the datasets we used in our project. Bolded names represent datasets not originally provided to us. SA represents Semantic Analysis, STS represents Semantic Textual Similarity, and Paraphrase represents Paraphrase Detection.

In our testing, we find that combining the original STS dataset with the SICK 2014 train data and combining the original SST data with 15k (evenly distributed) entries from the Rotten Tomatos dataset results in optimal performance. Table 2

Task	Final Train Set	Size
SA	SST Train + Rotten Tomatos (15k)	23,544
Paraphrase Detection	Quora Train	141,506
STS	SemEval SST Train + SICK2014 Train	10,540

Table 2: Final datasets selected for training.

Evaluation method To evaluate the 4 datasets provided to us, we will be evaluating them using a few different metrics. As described in the project spec, for datasets with binary labels (SST, CFIMDB, Quora), we will utilize accuracy as our evaluation metric. For datasets with non-binary labels (STS), we will utilize the Pearson correlation of the true similarity values against the predicted similarity.

Baselines As a lower bound, we have a pre-trained, fixed BERT model with only a single additional linear layer to the head (or final step) of the BERT model. We find that this model returns results of .411 for SST, .675 for Paraphrase Detection, and .272 for STS with an average accuracy of .453.

For an upper bound, we will also compare our scores for the Quora (QQP) and SemEVAL STS dataset (STS-B) against a default BERT base model’s (with fine-tuning) result from the original BERT paper Devlin et al. (2018) at Figure A. Because the datasets given to us are not the exact same as the ones in the GLUE benchmark, the results from Devlin cannot be used as an exact comparison.

Experimental details We experiment with different model architectures, including the BERT-Base model, which uses [CLS] token as sentence embedding and single linear downstream heads, and also Sentence-BERT as described in Section 4. We add dropout with probability 0.1 to the sentence embeddings. We also experimented with different adaptation modules, samplers, and datasets. We train the model with both finetune mode (in which we tune all parameters) and pretrain mode (in which we only tune adaptation modules and downstream heads).

We sample each task data at each step utilizing the sampling methods described in our approach 4. We optimize with AdamW optimizer, and use Noam scheduler with ten percent of warmup steps. The Noam scheduler increases learning rate linearly during warm up steps, and decreases it thereafter proportionally to the inverse square root of the step number. We tune the learning rate for backbone model, adaptation module, and downstream classifier separately. We use a learning rate 1e-5 for backbone model, and choose the best learning over range [1e-5, 1e-3] for adaption modules and downstream heads. We run experiments with batch size 16, number of steps per epoch 2400, and number of epochs 25. We save the model with best average validation across over three tasks each epoch.

Results

Result of SST and CFIMDB Table 3 below shows our result for minBERT implementation.

Finetune/Pretrain	SST	CFIMDB
Finetune	53.8	96.7
Pretrain	40.8	78.0

Table 3: Results of default minBERT implementation. The evaluation metric is accuracy.

Parameter/Performance Trade-Off for Multi-Task Fine-Tuning Table 4 and Figure 3 shows result of multi-task learning for different methods. We experimented with two backbone models: Base BERT and Sentence BERT, three adaptation modules: PAL, prefix, and adapter, and two training modes: pretrain and finetune. Base BERT uses the [CLS] token as sentence embedding and double linear downstream heads. Sentence BERT as described in section 4 uses mean pooling as sentence embedding and cosine similarity for similarity task. In pretrain mode, only task specific parameters are tuned, while in finetune mode, all parameters are tuned. To make the adaptation modules sizes comparable, we set prefix length 10, PAL size 204, and adapter size 64.

Sentence-BERT outperforms Base BERT in the paraphrase and similarity task. It achieves 32.2% improvement in similarity task. Our result shows that Sentence-BERT learns more semantically meaningful embeddings. Further investigation of Sentence-BERT’s features is in in section 6.

Adaptation modules make about 8% to 10% average score improvement in pretrain mode. In finetune mode, it only makes 3.7% improvement with Base BERT, and 0.6% percent improvement with Sentence-BERT. Because in finetune mode, the benefits of these adaptation modules changing attention activation will be alleviated by tuning the weights directly. Note that the main benefits of these adaptation modules is parameter efficiency. These adaptation modules in pretrain mode achieve

comparable results to full fine-tuning with less than 3% to 9% of trainable parameters. For example, *Sentence-BERT + Prefix* in pretrain mode achieves 65.6% average score with only 3% of trainable parameters compared to *Sentece-BERT* in fine-tune model, which has average score 69.4%.

Among the three adaptation modules, we found generally prefix has the best performance, and PAL has the second. We infer that the reason is prefix and PAL is parallely inserted into BERT layers, while adapter is sequentially inserted into BERT layers. Prefix is the most parameter efficient amongst the three methods. PAL is more parameter efficient than adapter, because it shares the projection layers across different layers.

Table 5 shows our best result on dev and test sets. We obtain our best result by slightly modifying the Sentence-BERT structure. We use double feed forward classification classifier that takes sentence embeddings u, v and $|u - v|$ as input for both paraphrase and similarity tasks, instead of calculating cosine similarity for similarity task. Further discussion of Sentence-BERT structure is in 6.

Finetune/Pretrain	Backbone	Adaptation	Trainable Param (M)	SST	Quora	STS	Avg
Pretrain	Base BERT	-	3.0	41.1	67.5	27.2	45.3
Pretrain	Base BERT	PAL	8.4	52.0	73.9	34.9	53.6
Pretrain	Base BERT	Prefix	3.2	50.5	72.3	36.7	53.2
Pretrain	Base BERT	Adapter	10.2	50.3	75.5	33.8	53.2
Pretrain	Sentence-BERT	-	2.4	45.0	72.2	49.5	55.6
Pretrain	Sentence-BERT	PAL	7.8	48.1	73.1	74.4	65.2
Pretrain	Sentence-BERT	Prefix	2.6	47.0	77.5	72.3	65.6
Pretrain	Sentence-BERT	Adapter	9.6	43.7	76.5	68.1	62.8
Finetune	Base BERT	-	112.4	50.0	81.5	43.7	58.4
Finetune	Base BERT	PAL	117.9	50.6	79.2	47.6	59.2
Finetune	Base BERT	Prefix	112.7	49.9	82.9	53.6	62.1
Finetune	Base BERT	Adapter	119.7	51.5	81.6	46.2	59.8
Finetune	Sentence-BERT	-	111.8	49.9	82.3	75.9	69.4
Finetune	Sentence-BERT	PAL	117.3	51.2	83.4	74.6	69.7
Finetune	Sentence-BERT	Prefix	112.1	50.5	81.9	77.5	70.0
Finetune	Sentence-BERT	Adapter	119.7	50.5	79.3	72.7	67.5
Ensemble x3				53.2	83.5	78.5	71.7

Table 4: Results of multi-task learning on SST, Quora, and STS datasets. SST and Quora are evaluated by accuracy. STS is evaluated by Pearson correlation. All scores are evaluated on dev set. *Pretrain* means only task-specific parameters are tuned; *Finetune* means all parameters are tuned. *Ensemble x3* is obtained by calculating weighted sum over logits of *Sentence-BERT + PAL*, *Sentence-BERT + Prefix*, and *Sentence-BERT* models.

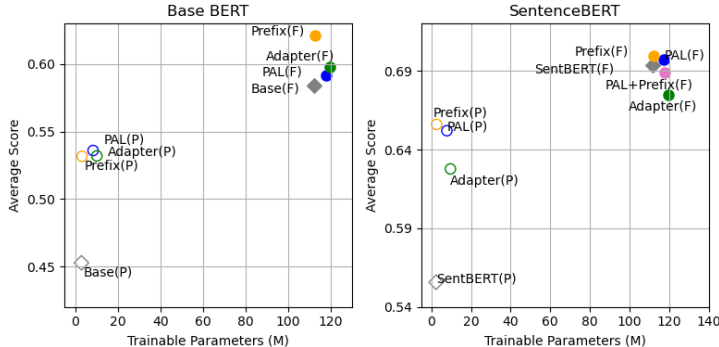


Figure 3: Performance v.s. trainable parameters of different methods. *P* means only tasks-specific parameters are tuned; *F* means all parameters are tuned.

	SST	Quora	STS	Avg
Dev	51.8	83.1	85.8	73.6
Test	52.5	83.3	84.9	73.6

Table 5: Best result on dev and test set with mean pooling and adding absolute difference term.

6 Analysis

Sentence-BERT feature analysis We found that Sentence-BERT outperforms Base BERT on paraphrase and similarity tasks. The three main differences between Sentence-BERT and Base BERT are:

- Uses cosine similarity instead of linear classifier for the similarity task.
- Uses mean pooling of sequence output instead of the [CLS] token for sentence embedding.
- Given sentence embeddings u and v , uses $(u, v, |u - v|)$ instead of (u, v) as input of sequence pair classifier.

Table 6 shows that the most important feature in Sentence-BERT is adding the $|u - v|$ term, which improves performance by 30.9% on similarity task. Taking meaning pooling and adding absolute difference term can further improve similarity task by 11.2%. The best model architecture uses mean pooling as sentence embedding and trains double feed forward classifiers that takes $(u, v, |u - v|)$ as input for both paraphrase and similarity tasks.

	SST	Quora	STS	Avg
Base BERT	50.0	81.5	43.7	58.4
+ cos-sim	48.6	81.5	51.7	60.6
+ mean pooling	51.9	81.2	47.1	60.6
+ $ u - v $	51.7	82.8	74.6	69.7
+ cos-sim + mean pooling	50.9	79.4	68.6	66.3
+ cos-sim + $ u - v $	48.9	80.1	55.5	61.5
+ mean pooling + $ u - v $	51.8	83.1	85.8	73.6
+ cos-sim + mean pooling + $ u - v $	51.0	82.2	72.3	68.5

Table 6: Analysis of Sentence-BERT features

Benefits of multi-task setting To analyze the capacity of our model, we fine-tune on the SST, Quora, and STS data sets separately. Table 7 shows that fine-tuning on the paraphrase task has better performance on the similarity task than fine-tuning on the similarity task itself. Note that in Sentence-BERT, we calculate the cosine-similarity between the mean pooled sequence output, so there is no trainable weight in the similarity downstream head. This result shows the importance of positive knowledge sharing across tasks. Recall that in the original design of prefix tuning and adapter, the backbone model is frozen, which prevents negative influence between tasks – but, this also eliminates positive information share between tasks. We infer this is the reason why the performance gap between our adaptation modules in pretrain mode and full fine-tuning is bigger than that reported in original prefix tuning and adapter papers.

Fine-tune dataset	SST	Quora	STS
SST	52.1	38.6	34.3
Quora	22.4	83.8	76.6
STS	18.6	41.7	66.1

Table 7: Result of fine-tuning on only single dataset, w/Sentence-BERT backbone.

Attention analysis To better understand how adaptation modules activate attention layers and how Sentence-BERT differs from Base BERT, we will analyze their attention maps. All attention maps in table 8 and table 9 are obtained from feeding a sentence selected from STS dev set: "Some guy sitting on a couch watching television." We draw the attention score after softmax, and average across attention heads.

Table 8 shows how prefix tuning activate different attention patterns for different tasks. We feed a sentence to the sentiment and similarity heads of Base BERT + prefix. Prefix tuning generates different attention map for the same sentence, which may enable the model to adapt to different tasks. In comparison, without adaptation modules, the attention map will be the same, because all tasks share the same backbone model.

Table 9 shows the attention patterns for Base BERT and Sentence BERT models for low, middle, and high layers. In general, Base BERT model will mostly attend at [CLS] token and have some diagonal attention at low layers, and attend at the end of sentence at high layers. Sentence BERT roughly has the same pattern at low layers, but its attention at high layers are distributed across tokens.

Last but not least, we would like to discuss why adaptation modules slightly gain improvement on Sentence-BERT in finetune mode. Table 9 shows that Sentence-BERT has very different attention map compared to Base-BERT. We infer that it resulted from mean pooling and absolute difference term for sentence pair classification. A possible reason may be the attention change in Sentence BERT is much larger than the attention change induced by adaptation modules.

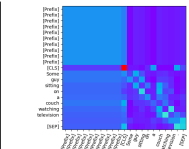
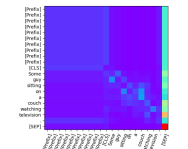
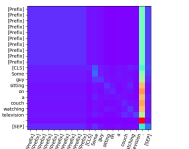
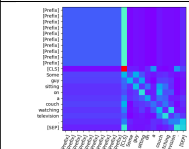
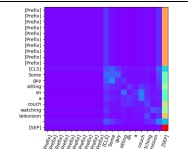
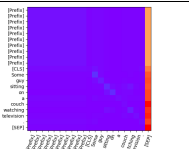
Downstream head	layer 1	layer 6	layer 12
SST			
STS			

Table 8: Attention map for Base BERT + prefix trained in pretrain mode by feeding the same sentence into SST and STS heads.

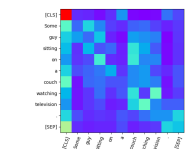
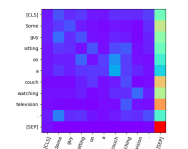
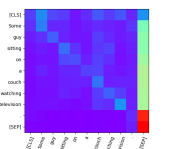
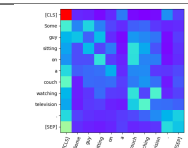
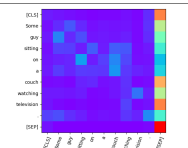
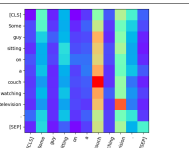
Backbone	layer 1	layer 6	layer 12
Base BERT			
Sentence-BERT			

Table 9: Attention map for Base BERT and Sentence-BERT trained in finetune mode.

7 Conclusion

Our goal was to introduce a BERT model that can build robust embeddings that perform well across a large range of different tasks, without having to finetune individual models for individual tasks. We

have successfully completed this task, through changes to the BERT-Base model's architecture as well as adding different adaptation modules and experimenting with how we train/what we train our model with.

We find that Sentence-BERT and mean pooling combined with using cosine similarity as our downstream classifier head for the STS task was critical to improving baseline scores. We showed an improvement of 32.2% in performance for the STS task when using Sentence-BERT with no adaptation modules as compared to Base-BERT when finetuning. This difference was even more dramatic in when pretraining with PALs – Sentence-BERT was able to improve by 39.5% over the base model with PALs.

References

- Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.
- Ruining He and Julian J. McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. *CoRR*, abs/1602.01585.
- Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp.
- Stefano Leone. 2021. Rotten tomatoes movies and critic reviews dataset.
- Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation.
- Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks.
- SemEval. 2014. [link].
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.
- Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning.

A Appendix

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
BERT _{BASE}	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
BERT _{LARGE}	86.7/85.9	72.1	92.7	94.9	60.5	86.5	89.3	70.1	82.1

Fig A. Baseline results for Base BERT from Devlin et al.

A.1 Dataset Information

The SICK-2014 dataset contains 10,000 English sentence pairs, each annotated with a semantic similarity score 0-5, following the structure of the SemEval STS benchmark dataset. SICK was created to fill the gap that the SemEval STS benchmark leaves – namely, that training examples deal with issues, such as identifying multi-word expressions, recognizing named entities or accessing encyclopedic knowledge, that semantic models are not expected to handle. Instead, SICK aims to capture only similarities on purely language and common knowledge level, without relying on domain knowledge. (SemEval, 2014)

The original Rotten Tomatoes dataset contains ~18k records representing movies, with each entry including the movie tile, description, genres, duration, director, actors, users’ ratings, and critics’ ratings. We then further cleaned the dataset to get ~634k individual review entries, where each entry is structured like the SST dataset with a review and sentiment score 0-4. Reviews were skimmed to ensure that all entries had a rating on a scale of 1 to 5, and were then mapped to a scale of 0 to 4. (Leone, 2021)

The Kindle review dataset contains 982,619 reviews from the Amazon Kindle store during the period of 2007 - 2014. We experiment with pulling reviews to create different sizes of datasets for training (80k, 40k, 20k) – in each cleaned dataset, reviews are mapped from 1 to 5 stars to a rating of 0 to 4 in accordance to the SA task. (He and McAuley, 2016)

A.2 Ablation Studies

Prefix length Longer prefixes may have better expressive power. Figure 4 shows that with the Base BERT backbone, the performance increases as the prefix length increases up to 15. With the Sentence-BERT backbone, the benefit of prefix isn't obvious.

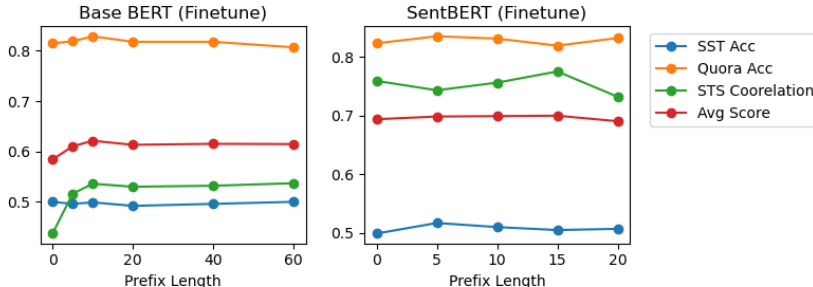


Figure 4: Prefix length v.s. performance for Base BERT and Sentence-BERT backbone in finetune mode.

Sampling method Table 10 shows the performance differences between choosing different sampling methods in our training pipeline. We see that annealed sampling is the method with highest overall performance.

Sampling Method	SST	Quora	STS	Avg
Round Robin	51.4	77.9	73.1	67.47
Proportional	50.7	81.0	75.5	69.07
Square Root	49.8	80.0	72.0	67.27
Annealed	49.8	84.3	74.0	69.37

Table 10: Result of different sampling methods on performance for fine-tuning. Prefix-tuning with a Sentence-BERT backbone was used for these tests.

Dataset Studies Table 11 shows the performance differences between choosing different training datasets in our training pipeline. We see that only adding the SICK dataset results in the highest overall performance. We tried to make the sentence length and label distribution of the additional Kindle and Rotten datasets similar to the SST dataset, but the domain difference and bias in score still make them a lot different than the SST dataset.

Datasets	SST	Quora	STS	Avg
Default	51.0	82.2	72.3	68.50
+ SICK	52.8	82.3	74.4	69.83
+ Rotten	46.8	82.1	75.9	68.27
+ Kindle	51.0	82.3	72.2	68.50
+ SICK, Rotten	50.0	82.3	77.1	69.80
+ SICK, Kindle	49.9	82.3	74.4	69.37

Table 11: Result of different training datasets on performance for finetuning. Sentence-BERT backbone was used for these tests.