Tuning Up BERT: A Symphony of Strategies for Downstream Tasks

Stanford CS224N Default Project

Nick Soulounias
Department of Computer Science
Stanford University
`s0ul@stanford.edu`

Abstract

This project explores the enhancement of foundation models like BERT, which are extensively pretrained primarily through self-supervision and are adaptable across various tasks through transfer learning. We focus on improving BERT's performance for three downstream tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. Our results indicate that cross-encoding improves the performance for paraphrase detection and semantic textual similarity significantly and that for the latter task multi-task learning further boosts performance. For the challenging sentiment analysis task, we observe significant overfitting, which we mitigate to some extent by generating synthetic data with GPT-3.5 and incorporating a SMART regularizer. Finally, we ensemble our multi-task and single-task models to achieve the 1st place in the dev set leaderboard with an overall score of 0.803 and a test score of 0.793.

1 Key Information to include

Mentor: Ryan Li • External Collaborators: No • Sharing project: No

2 Introduction

In recent years, the field of Natural Language Processing (NLP) has been profoundly transformed by the advent of foundation models. These large language models, such as the BERT Devlin et al. (2018), have set new standards for a range of NLP tasks due to their deep contextual understanding and adaptability. Extensive pretraining on massive and diverse corpora allows BERT to develop a broad understanding of language dynamics before being fine-tuned to specific tasks. This approach not only improves model performance but also significantly reduces the need for task-specific data, making high-quality NLP capabilities more accessible. Our project taps into this potential by adapting the BERT model for sentiment classification, paraphrase detection, and semantic textual similarity.

In this paper, we investigate both single-task and multi-task learning, explore different techniques for sentence-pair tasks, mitigate overfitting by generating synthetic data and optimizing a regularized optimization objective, compare different task sampling strategies for the multi-task problem setting and implement model ensembling to further improve performance. Through iterative development based on our experiments and insights, our final model achieves a significant performance boost for all three downstream tasks compared to the baseline.

3 Related Work

Devlin et al. (2018) introduced the BERT model architecture and proposed handling sentence pairs through a technique known as cross-encoding. In this setup, BERT processes a pair of sentences as a single input, separated by a special token ([SEP]), which enables the model to jointly analyze the sentences. Reimers and Gurevych (2019) proposed a more compute-efficient alternative for handling sentence pairs by utilizing a siamese network structure, significantly reducing the required time and computational resources by generating sentence embeddings that can be compared using cosine similarity.

Jiang et al. (2019) addressed issues of model overfitting and knowledge forgetting when fine-tuning pretrained language models like BERT and RoBERTa on relatively small downstream tasks. They introduced a regularization penalty to ensure the model's outputs do not drastically change with small perturbations in the input and employed a trust-region method to prevent aggressive updates, enhancing performance across multiple NLP benchmarks.

Stickland and Murray (2019) tackled the challenges of multitask learning when dealing with tasks of varying dataset sizes, noting that smaller datasets often lead to model overfitting, while larger datasets may result in underfitting. They developed a method called annealed sampling, which adjusts the probabilities for selecting tasks based on their dataset sizes and strategically introduces tasks during the training process to optimize learning outcomes.

Li et al. (2023) explored the generation of synthetic data using large language models (LLMs) for text classification tasks, assessing the efficacy of this data in training models like BERT and RoBERTa. Their research indicates that while synthetic data can partially substitute real data in training scenarios, its effectiveness varies significantly with the task's nature, particularly in terms of subjectivity.

Drawing inspiration from these works, our project seeks to enhance the capabilities of BERT across three specific downstream tasks, aiming to significantly refine its adaptability and accuracy in diverse NLP applications.

4 Approach

4.1 MinBERT Model Architecture

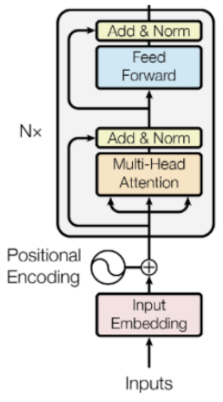

Figure 1: The architecture of the minBERT base model

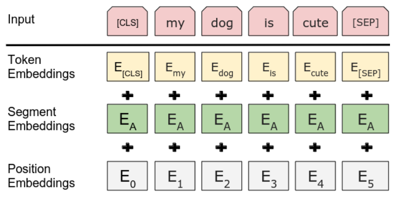

Figure 2: The embeddings for single-sentence inputs.

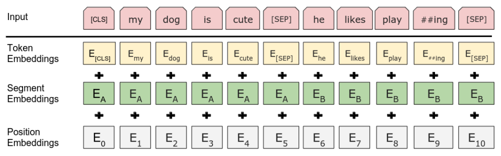

Figure 3: The embeddings for sentence-pair inputs.

The minBERT model is a minimal implementation of the BERT encoder-only Transformer Devlin et al. (2018). The input sentence is tokenized with a WordPiece tokenizer Wu et al. (2016) with a vocabulary of 30,000 tokens. The tokenizer adds a special [CLS] token at the beginning of the sentence and a special [SEP] token at the end. The output is a list of token ids with a maximum length of 512 and special [PAD] tokens appended if necessary after the last [SEP] token. Word pieces that are not included in the vocabulary are tokenized as the [UNK] special token.

The model consists of an embedding layer, followed by 12 Transformer layers (Fig. 1). The embedding layer takes as input a sequence of tokens ids and maps each of the tokens to a 768-dimensional vector. This representation is obtained by summing the corresponding learned token embeddings, the absolute position embeddings and the learned segment embeddings, which are relevant only when tokenizing sentence pairs for cross-encoding (Fig. 2). The embeddings are normalized with layer normalization and regularized by applying dropout.

Each transformer layer consists of two sub-layer modules, multi-head attention and the feed-forward network. Multi-head attention allows for each token to attend to multiple representations of any token in the sequence, other than the [PAD] ones, and update its information. The feedforward network applies a linear transformation to every token representation with a GeLU activation function, followed by another linear transformation. Each sub-layer employs a residual connection, followed by dropout regularization and layer normalization.

Given that the model is fine-tuned for three tasks, we add a task-specific head per task, and use the output embedding of the [CLS] token as the input for every head. For the 5-way sentiment classification task, the task-specific head is a linear layer that outputs 5 logits. During training, we minimize the cross-entropy loss between the predictions and the ground-truth labels. For the paraphrase detection and the semantic contextual similarity tasks, which are both sentence-pair tasks, we consider both a bi-encoding and a cross-encoding approach and differentiate the task-specific head architecture accordingly.

4.2 Bi-encoding, Cross-encoding

For both the semantic textual similarity and paraphrase detection tasks, the model's input is a sentence-pair. Our initial implementation of minBERT was limited to to single-sentence inputs. To this end, we consider two strategies to extend the model for the sentence-pair tasks: bi-encoding Reimers and Gurevych (2019) and cross-encoding Devlin et al. (2018).

The bi-encoding approach fuses the information of the two sentences after they have been independently processed by the base BERT model. Specifically, we obtain for each sentence the [CLS] output and apply a linear projection to an embedding space with the same dimensionality. We compute the cosine similarity between the embeddings of the two sentences. For the paraphrase task, if we apply the sigmoid function to the cosine similarity score we get the probability that the sentences are a paraphrase of each other and we minimize the binary cross-entropy loss between the predicted logits and the ground-truth labels. For the semantic textual similarity score, we scale the cosine similarity score in the [0, 5] range and minimize the mean squared error between the predicted and the ground-truth score.

Cross-encoding does not fuse the information of the two sentences after processing them individually. Instead the two sentences are tokenized together and an intermediate special [SEP] token is added between them. Furthermore, the appropriate learned segment embedding is used in the embedding layer depending on whether a token belongs in the first or the second sentence (Fig. 3). This approach enables tokens to attend to information from both sentences and, as a result, the [CLS] output embedding is a representation of the sentence-pair. For the paraphrase detection head, the task-specific head contains a linear projection to 2 logits. By applying the softmax function to these logits, we can obtain the probabilities that the sentence-pair is a paraphrase or not. We optimize the cross-entropy loss between the predicted logits and the ground-truth labels during training. For the semantic textual similarity task, we apply a linear projection which outputs the predicted score. During training, we minimize the mean squared error between the predicted and the ground-truth scores, as previously. We clip the predicted scores to be within the [0, 5] range only as a post-processing step during evaluation.

4.3 Synthetic data generation

When minimizing the training loss for the SST-5 dataset, the loss on the development holdout set first decreases and then increases, indicating overfitting on the training set. Collecting new data could mitigate this, but manually assessing and labeling movie reviews for sentiment analysis requires significant human effort. Instead, autoregressive large language models like GPT can generate synthetic data in a scalable way.

Initially, we prompted the GPT model to generate both a movie review sentence and the appropriate sentiment label. However, to ensure the synthetic data's usefulness, the model's assigned labels should align with the labels of the SST-5 annotators. We tested the most capable GPT model, GPT-4 Turbo, on the SST-5 development set and found its accuracy to be only 0.548. Preliminary investigations for GPT-3.5 Turbo and GPT-4o on a subset of the development set indicated a further decrease in performance. This low accuracy suggests a misalignment with the SST-5 dataset annotators, leading us to explore other options.

Fine-grained sentiment classification can be challenging due to noisy labels, but paraphrasing a given movie review is substantially easier. We qualitatively investigated paraphrased sentences generated by GPT-3.5 Turbo and observed high quality, with minimal overlap in words with the original sentences. We prompted the model to maintain the same sentiment in the paraphrased sentence and assign the same label. Consequently, we were able to use GPT-3.5 Turbo, which is 10x cheaper than GPT-4 Turbo, to generate a high-quality synthetic dataset with 100K new sentences, whose labels are consistent with the real data.

4.4 SMART Regularization

Overtraining on the SST-5 dataset leads to overfitting on a training set with noisy labels and poor generalization on the holdout development set. To this end, we investigate whether finetuning with regularized optimization can improve the results. Specifically, we incorporate the Smoothness Inducing Adversarial Regularization technique(SMART) Jiang et al. (2019) to our optimization procedure. SMART aims to keep the model's output consistent when small perturbations are applied to the embeddings from the first layer of the language model, thereby improving the model's ability to generalize to unseen datapoints.

Concretely, the optimization objective, which we aim to minimize, includes a weighted smoothness-inducing adversarial regularization loss $R_s(\theta)$ in addition to the standard task-specific loss function $L(\theta)$:

$$\theta^* = \arg\min_\theta L(\theta) + \lambda_s R_s(\theta)$$

where $\lambda_s > 0$ is a hyperparameter and the smoothness-inducing regularizer $R_s(\theta)$ is defined as:

$$R_s(\theta) = \frac{1}{n}\sum_{i=1}^{n} \max_{\|\tilde{x}_i - x_i\|_p \leq \epsilon} l(f(\tilde{x}_i;\theta), f(x_i;\theta))$$

The number of data point of the target task is denoted as n, x_i and $\tilde{x}_i$ correspond to the embeddings and the perturbed embeddings of the i-th datapoint, respectively, ϵ is a hyperparameter, and l is defined as the symmetric KL-divergence for the sentiment classification and the paraphrase detection tasks and as the mean-square error for the semantic textual similarity task. The computation of $R_s(\theta)$ involves a maximization problem, which is solved by projected gradient ascent with a step-size η hyperparameter. We note that in the original paper, the authors also included a trust-region-type regularization named Bregman Proximal Point Optimization, which we omit.

4.5 Multitask Learning

Multi-task learning optimizes for all three downstream tasks simultaneously. The base model layers are shared across tasks, while the heads are task-specific. For each parameter update step we select one task randomly, sample a batch from the corresponding dataset and minimize the appropriate loss. We consider two strategies to select the tasks, uniform sampling and annealed sampling.

Uniform sampling assigns the same probability to every task, i.e. $p_i = \frac{1}{3}$. Given that different tasks may have distinct optimal parameters, optimizing for one task for multiple gradient descent steps may lead to a suboptimal subspace in the parameter landscape for the other tasks and escaping that subspace may prove challenging. Thus, this sampling strategy does not prioritize one task over another in order to learn a representation suitable for all three downstream tasks.

The main motivation behind annealed sampling Stickland and Murray (2019) is the significant difference between dataset sizes. Uniform sampling exhausts the smaller datasets while having processed only a small subset of the larger ones. Our hypothesis was that this could lead to overfitting on the data-poor tasks, while undertraining on the data-rich ones. Annealed sampling addresses this issue by prioritizing tasks with more data. Specifically, task i is sampled with probability p_i, such that:

$$p_i \propto N_i^{1-0.8\frac{e-1}{E-1}}$$

where N_i is the dataset size, e the current epoch and E the total number of epochs. We consider the generated synthetic data for the SST-5 dataset as a form of data augmentation on the original sentences of the dataset that are paraphrased, and thus N_i always corresponds to the number of real datapoints in the dataset. With annealed sampling, data-rich tasks are selected more often during the first epochs and with more equal probability towards the end of training.

5 Experiments

5.1 Data & Evaluation

We finetune and evaluate the model's downstream performance using three datasets. The Stanford Sentiment Treebank Dataset (SST-5) Socher et al. (2013) consists of 11,855 phrases extracted from Rotten Tomatoes movie reviews. Each phrase is labelled as negative, somewhat negative, neutral, somewhat positive, or positive. The Quora Question Pair Dataset (QQP) Fernando and Stevenson (2008) is used for paraphrase detection and consists of 404,298 two-sentence question pairs that are labelled as either a paraphrase or not. The SemEVAL STS Benchmark Dataset (STS) Agirre et al. (2013) consists of 8,628 two-sentence pairs of varying similarity on a scale from 0 (unrelated) to 5 (equivalent meaning).

When evaluating the performance of the model on the SST-5 and QQP datasets, we report the accuracy. For the STS dataset, we use the Pearson correlation and the overall score is computed as the average between the accuracies and the normalized Pearson correlation between 0 and 1.

5.2 Experimental details

Throughout our experiments, we set the dropout probability to 0.3 and select the batch size to be 16. All models were trained using the implemented AdamW optimizer with a learning rate of 1e-5 and coefficient parameters $(\beta_1, \beta_2) = (0.9, 0.999)$ with no weight decay applied. In the single-task setting, we train the model for the paraphrase detection task for 25 epochs and 2500 parameter update steps per epoch. For the sentiment classification task and the semantic textual similarity task, the model is trained for 150 epochs with 50 parameter steps per epoch. In the multi-task setting, all the models are trained for 25 epochs with 1000 parameter update steps per epoch. Each model is evaluated on the holdout development set once per epoch and we save the parameters that attained the best score. All the methods that were described in the previous section were implemented from scratch without using any external libraries.

5.3 Results

5.3.1 Sentence-pair encoding

	Dev QQP	**Dev STS**
Sentence-pair Encoding	Accuracy	Pearson Correlation
Bi-encoding	0.605	0.544
Cross-encoding	**0.905**	**0.896**

Table 1: The dev-set results for bi-encoding and cross-encoding sentence-pairs.

In our baseline implementation for the sentence-pair tasks we followed the bi-encoding strategy proposed by Reimers and Gurevych (2019), which only required minimal changes in the provided codebase. However, as the results in Table 1 indicate, the cross-encoding strategy Devlin et al. (2018) resulted in a substantial improvement for both the QQP and the STS task.

These results indicate that the late-fusion of sentence information, employed by the bi-encoding method, is not expressive enough to learn intricate dependencies between the two sentences. Instead, cross-encoding, which allows the model to attend directly to interactions between elements of both sentences throughout the entire network, can capture deeper and more nuanced relationships.

Following these results, we decide to use the latter encoding method for both sentence-pair tasks in all subsequent experiments.

5.3.2 Synthetic data

During our experiments with the SST-5 task, we observed a notable trend of overfitting within the training dataset, as highlighted in Figure 4. In an attempt to address this challenge, we incorporated a combination of real and synthetic data into our training regimen. This adjustment resulted in a modest performance enhancement, improving accuracy by 1%. Encouraged by their positive impact, we decided to continue the use of synthetic data in the training phase for all subsequent experiments. Nevertheless, we have to note that training on both synthetic and real data did not completely mitigate overfitting.

Training Data	**Dev SST-5** Accuracy
Real data	0.530
Real + Synthetic data	**0.540**

Table 2: The effect of adding the synthetic data on the SST-5 training set.

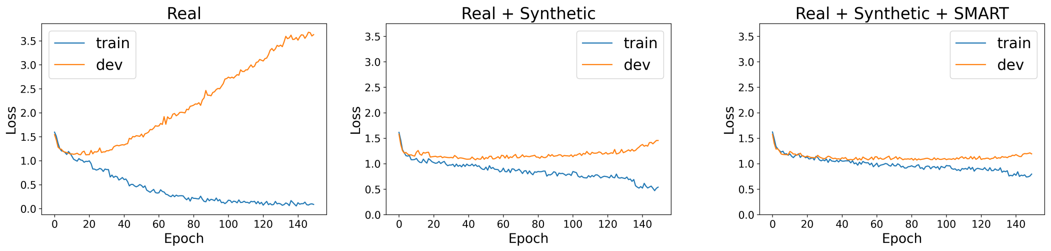

Figure 4: The effect of synthetic data and SMART regularization on SST-5 overfitting.

5.3.3 SMART regularization

Motivated by the persisting issue of overfitting for the SST-5 dataset, we integrated the SMART regularizer in our optimization objective. Furthermore, following the success of Jiang et al. (2019) in QQP and the improvements of past CS224N projects Chijioke et al. (2024); Jack and Febie (2024) in STS by incroporating SMART, we also investigate its applicability in our case. Following the notation of Jiang et al. (2019), we use their recommended hyperparameters and set the perturbation size $\epsilon = 10^{-5}$ and $\sigma = 10^{-5}$, the adversarial learning rate $\eta = 10^{-3}$, the norm $p = \infty$ and the adversarial steps $T_{\bar{x}} = 1$. We experiment with the smoothing coefficient λ_s and report our results in Table 3. SMART improved the accuracy on the SST-5 development set by 0.9%, but we did not get any improvement for the other tasks. We verify the beneficial effect of SMART regularization to overfitting in Figure 4.

SMART	**Dev SST-5** Accuracy	**Dev QQP** Accuracy	**Dev STS** Pearson Correlation
(1) $\lambda_s = 0$	0.540	**0.905**	**0.896**
(2) $\lambda_s = 1$	0.543	0.904	0.882
(3) $\lambda_s = 3$	**0.549**	0.882	0.894

Table 3: The effect of incorporating SMART regularization for different smoothing coefficients λ_s.

5.4 Multi-task training

We began by investigating task sampling strategies within a multi-task framework. According to our results in Table 4, uniform sampling demonstrates strong performance in the STS dataset and improves the corresponding single-task model, while annealed sampling outperforms uniform sampling in both the SST and QQP datasets. Given SMART's success in single-task learning for the SST dataset, we integrated SMART with $\lambda_s = 3$ in our multi-task setting. However, this regularization adversely affected the QQP dataset, preventing any overall score improvement.

We theorize that uniform sampling underperforms relative to annealed sampling for the QQP task due to insufficient training duration on QQP data. Extending training epochs poses a challenge, as it leads to overfitting on the SST-5 task. To address this, we initialized BERT with the weights of the single-task model that previously achieved the best performance on QQP. While this approach achieved the highest accuracy on QQP, it led to suboptimal performance on the SST-5 and STS tasks, primarily because the initial parameters did not lead to a landscape where the model could generalize well for the SST-5 dataset.

Multi-task training strategies	**Dev SST-5** Accuracy	**Dev QQP** Accuracy	**Dev STS** P/S Corr	**Dev Overall** Score
uniform sampling	0.518 (0.527)	0.881 (0.881)	**0.900** (**0.902**)	0.783
annealed sampling	**0.534** (**0.534**)	0.895 (0.896)	0.888 (0.892)	**0.791**
uniform sampling + SMART	0.529 (0.529)	0.875 (0.875)	**0.900** (**0.902**)	0.784
annealed sampling + SMART	0.530 (0.530)	0.885 (0.894)	0.894 (0.896)	0.787
uniform sampling + QQP initialization	0.526 (0.526)	**0.901** (**0.905**)	0.895 (0.900)	**0.791**

Table 4: The effect of different training strategies in the multi-task setting. The scores in parentheses correspond to best task-specific scores of the model, while the other scores correspond to best overall model.

5.5 Model Ensembling

Finally, we chose to strategically ensemble some of the best single-task and multi-task models that we had previously trained to further boost performance across the three tasks. For the classification tasks, we ensembled the models by computing and averaging their assigned probabilities, and selected the most probable class as the ensemble prediction. For the regression task, we simply averaged the outputs of the models to compute the final prediction. We report our best results in Table 6 when using a single model for all three tasks, at most one different model per task, and a model ensemble comprising 1-4 models per task. It is important to note that, contrary to our expectations, ensembles for the SST-5 task with more than one model led to worse results on the corresponding dev set.

Results	**Dev SST-5** Accuracy	**Dev QQP** Accuracy	**Dev STS** P/S Corr	**Dev Overall** Score
1 model / 3 tasks	0.534	0.895	0.888	0.791
1 model / 1 task	0.549	0.905	0.902	0.802
1 ensemble / 1 task	0.549 (1 model)	0.910 (3 models)	0.905 (2 models)	0.803

Table 5: The effect of ensembling on the dev set performance.

We evaluated the model ensembles that achieved the best results on the dev set for the hidden test set:

	Test SST-5	**Test QQP**	**Test STS**	**Test Overall**
1 ensemble / 1 task	0.519 (1 model)	0.910 (3 models)	0.900 (2 models)	0.793

Table 6: Our final results on the hidden test set evaluation.

We observed that the dev set performance genrelizes to the test set for the QQP and the STS tasks and there is some degradation for the SST-5 task, indicating that using an ensemble of models for this task would have been helpful, despite the potentially lower dev performance.

6 Analysis

6.1 Sentiment Analysis

The confusion matrix in Figure 5 provides insights into the challenges of sentiment classification, particularly in terms of the misclassifications occurring between adjacent sentiment categories. These misclassifications underscore the inherent subjectivity embedded in sentiment analysis.

To explore this phenomenon further, we also conducted a qualitative investigation into specific instances where the model's predictions diverge from the originally assigned labels. Our findings are presented in Table 7, where we highlight cases where the model's predicted sentiments appear to be more aligned with the context of the text than the assigned labels. Such discrepancies suggest the presence of noisy labels within the training data, which is due to the subjective nature of the task and the diverse interpretations of the annotators. We hypothesize that the inclusion of such noisy labels in the training data is one of the reasons why employing SMART regularization led to improved model performance, even when a substantial amount of synthetic data was available. In this way, the model is discouraged from overfitting to the noisy training data and can generalize better on unseen datasets.

6.2 Paraphrase Detection

We visualize the confusion matrix for the QQP development set in Fig. 5. The figure demonstrates the robust performance of the model, with the misclassifications being balanced between false positives and false negatives. We inspect qualitatively the model's failure cases and present some representative examples in Table 8. We identify that the model often predicts a false positive when the sentences have similar words or phrases that do not carry the same meaning in different contexts (Ex. 1). On the other hand, the model's false negatives occur when some secondary information are excluded from one sentence (Ex. 2). We argue that in these cases labelling the sentences as a paraphrase or not is subjective. Additionally, replacing a word with a synonym (Ex. 3) or changing the sentence structure (Ex. 4) may also result in false negatives.

6.3 Textual Semantic Similarity

We visualize the model's behavior for the STS task as a scatter plot in Figure 6, which indicates a strong positive correlation between the predicted and true labels. The data points are more spread for the mid-range scores, which is expected since this sentences have more subtleties and assigning an accurate score as label can be challenging. The model predictions at the extreme values are very accurate, which means that when the model identifies two sentences as completely unrelated or related, this is indeed the case. However, the vice-versa does not always hold; not all completely related or unrelated sentences are identified as such. As the examples 1 and 2 suggest in Table 9, using abbreviations of words causes the model to misidentify two related sentence. Further, the model is misled when the two sentences have many similar words and focus on the same general topic, the inverse phenomenon is observed; the model considers somewhat related two semantically distinct sentences (Ex. 3 and 4).

7 Conclusion

In this project we fine-tuned BERT model for specialized NLP tasks, achieving significant improvements in paraphrase detection, sentiment analysis, and semantic textual similarity. Techniques such as cross-encoding have significantly enhanced the model's ability to analyze complex sentence relationships, while the integration of synthetic data and SMART regularization has effectively mitigated overfitting in sentiment analysis. Furthermore, we showcased that multi-task learning can lead to positive transfer for some tasks and further boost their single-task performance.

Despite these advancements, challenges remain, particularly in optimizing multitask learning to balance the specific needs of all tasks simultaneously. Future research could focus on refining our multitask strategies by integrating more advanced optimization methods, such as PCGrad Yu et al. (2020) and more efficient ensembling strategies that do not require training multiple independent models Lee et al. (2022), enhancing both the efficiency and accessibility of foundation models in NLP.

Sentence	Prediction	Label
there ’s not enough here to justify the almost two hours .	0	1
a marvel like none you ’ve seen .	4	3
michael gerbosi ’s script is economically packed with telling scenes .	3	4
cool ?	2	3
half submarine flick , half ghost story , all in one criminally neglected film	1	2

Table 7: A qualitative investigation of the SST-5 results.

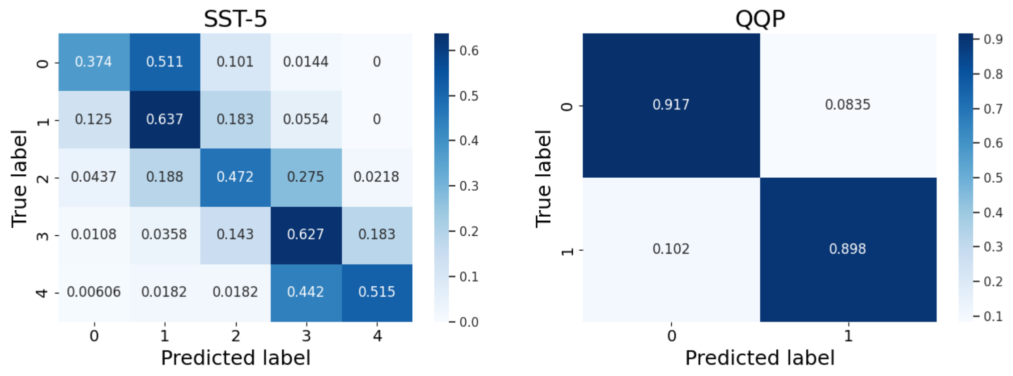

Figure 5: The normalized confusion matrices for SST-5 and QQP.

8 Ethics Statement

In this project, we demonstrate that large language models can be effectively fine-tuned for specific downstream tasks. However, deploying these models in production requires a comprehensive understanding of their broader societal implications. The potential automation of tasks traditionally performed by humans could significantly increase unemployment and concentrate wealth among a small number of individuals. It is imperative for governments and organizations to evaluate these risks and actively develop strategies to support the transition and reskilling of affected individuals. Additionally, apart from assessing the model’s average accuracy, fairness across protected groups, extensive testing and transparency about the model’s limitations are essential to ensure ethical deployment.

Our best results were achieved through model ensembling, which involves training multiple models. While this approach enhances performance, it also increases computational and energy demands, consequently elevating the carbon footprint. Moreover, the reliance on extensive GPU resources exacerbates the divide between GPU-rich and GPU-poor entities. Developing resource-efficient ensembling techniques is critical to ensure equitable access to advanced machine learning technologies and to minimize their environmental impact.

	Sentences	**Pred**	**Label**
1	what are the uses of arduino ? what is the software used in arduino ?	1	0
2	how much will it cost for studying ms in germany how much does it cost for an ms in mechanical in germany in 2016 -2018 in inr ?	0	1
3	how do i write an android application ? how do i create an android application ?	0	1
4	will american parents in the future going to be the most paranoid parents in the world ? will american parenting in the future going to be very overprotective ?	0	1

Table 8: A qualitative investigation of the failure cases in QQP.

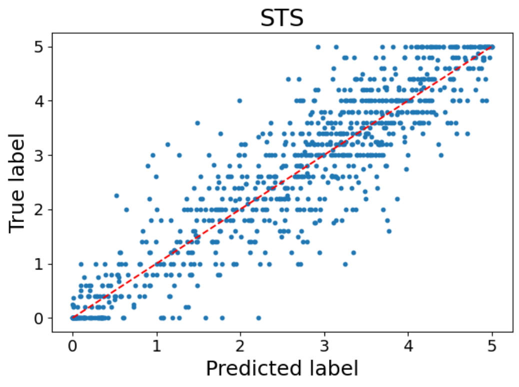

Figure 6: The scatter plot analyzes the error of the model in the STS dev set.

	Sentences	**Pred**	**Label**
1	carney sets high bar to change at boe carney sets high bar to changes at bank of england	2.93	5.00
2	mitt romney wins maryland gop primary romney wins maryland republican primary	3.71	4.80
3	live blog: ukraine in crisis live blog: iraq in turmoil	1.07	0.00
4	3 killed , 4 injured in los angeles shootings five killed in saudi arabia shooting	1.09	0.00

Table 9: A qualitative investigation of the outliers in STS.

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.

Mgbahurike Chijioke, Mlauz Iddah, and Ocran Kwame. 2024. Jack of all trades, master of some: Improving bert for multitask learning. *CS224N*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Samuel Fernando and Mark Stevenson. 2008. A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th annual research colloquium of the UK special interest group for computational linguistics*, pages 45–52.

Le Jack and Lin Febie. 2024. Bert and beyond: A study of multitask learning strategies for nlp. *CS224N*.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2019. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *arXiv preprint arXiv:1911.03437*.

Yoonho Lee, Huaxiu Yao, and Chelsea Finn. 2022. Diversify and disambiguate: Learning from underspecified data. *arXiv preprint arXiv:2202.03418*.

Zhuoyan Li, Hangxiao Zhu, Zhuoran Lu, and Ming Yin. 2023. Synthetic data generation with large language models for text classification: Potential and limitations. *arXiv preprint arXiv:2310.07849*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y 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.

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *International Conference on Machine Learning*, pages 5986–5995. PMLR.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836.