BERT-icus, Transform and Ensemble!

Stanford CS224N Default Project

Maya Czeneszew
Dept. of Computer Science
Stanford University
mayacz22@stanford.edu

Helen He
Dept. of Computer Science
Stanford University
helenahe@stanford.edu

Sidra Nadeem
Dept. of Computer Science
Stanford University
sidranem@stanford.edu

Abstract

"Transformers, more than meets the eye!" Transformers have revolutionized natural language processing (NLP) with models like BERT, but they still face challenges such as overfitting and lacking nuanced language understanding. We explore these challenges using minBERT over three human language tasks: sentiment analysis, paraphrase detection, and textual similarity tasks. Techniques such as smoothness-inducing adversarial regularization and cosine-similarity fine-tuning augment minBERT's performance. Ensembling methods further enhance results, with weighted "expert" averaging yielding optimal outcomes. Our findings contribute to NLP discourse, emphasizing domain expertise and task-specific fine-tuning for effective model optimization, while proposing future avenues for exploration. Through these efforts, we advance the understanding and application of NLP techniques, paving the way for more robust language models to roll out.

1 KEY INFORMATION

Mentor: Annabelle Wang | External Collaborators: None | Sharing Project: No

2 INTRODUCTION

"Transformers, more than meets the eye. Transformers, attention in disguise." These iconic words, evoking memories of a popular cartoon theme, resonate deeply in the realm of artificial intelligence. In recent years, the field of natural language processing (NLP) has experienced a surge in innovation, propelled by seminal works such as "Attention Is All You Need" by Vaswani et al. This groundbreaking research introduced transformers, revolutionizing machine understanding of human language and giving rise to models like BERT, Transformer-XL, and of course, the household name GPT.

Our project focuses on exploring the capabilities of one such model: miniature BERT (minBERT), aiming to address fundamental NLP tasks including sentence sentiment classification, paraphrase detection, and semantic textual similarity. Despite their remarkable potential, transformers have inherent limitations, such as susceptibility to overfitting and overlooking nuances in language.

To overcome these challenges and enhance minBERT's performance, we employ advanced ML techniques, including smoothness-inducing adversarial regularization, cosine-similarity fine-tuning, and ensemble methods. By integrating these techniques, our goal is to improve minBERT's robustness, flexibility, and adaptability, enabling it to capture a broader range of linguistic patterns. Through this effort, we aim to contribute to the ongoing evolution of NLP models, advancing natural language understanding systems.

3 RELATED WORK: BERT, A RENAISSANCE MODEL

BERT multitask learning in particular can leverage information from multiple related NLP tasks to improve the performance of the overall model. Similar to how humans transfer their knowledge from one task to another, we can use the same principle when training models for downstream NLP tasks (Zhang and Yang, 2017). Prior research has also shown that multitasking fine tuning before single-task fine tuning improves BERT's accuracy on the development set (Mienye and Sun, 2020).

However, aggressive fine-tuning can lead to overfitting with downstream tasks (Kuhn and Johnson, 2013). We therefore focus on reducing the chance for overfitting and increasing flexibility when tested with new data to improve our minBERT model with the following approaches.

Our first goal is to reduce overfitting across all three tasks in order for our model to generalize better to shared learning tasks. This is achieved through implementing two versions of the smoothness-inducing adversarial regularization technique, the first being directly from the work of Jiang et al., and the second being a hybrid approach that incorporates parameter perturbation techniques.

Our second goal is to explore the synergies and interplay between fine-tuning single tasks and multiple tasks, given the distinct advantages of each approach. For this, we experiment with single-task fine-tuning, multitask learning, and ensemble learning.

In particular, we're intrigued by the potential of ensemble learning. Often shadowed by its more popular counterpart multitask learning, ensemble learning involves training several different models separately, and then combining model predictions through various statistical methods. Ensemble learning has been shown to improve results on unbalanced datasets, prompting us to use this approach due to the significantly greater size of the Quora Dataset (Mienye and Sun, 2022). Another benefit of ensemble learning is more diversified knowledge, since the technique incorporates a range of base learners that allow for a more varied error pattern and a more robust aggregate prediction (Bian et al., 2020).

4 APPROACH

4.1 The Basic minBERT

First, we implemented a pre-trained minBERT model as per the comprehensive project guidelines (Devlin et al., 2018). Features include a multiheaded self-attention module and an Adam optimizer.

To mitigate common mishaps such as overfitting to training data and overlooking subtle linguistic nuances, we have added the following methods specifically aimed at enhancing model flexibility and generalization.

4.2 Smoothness-Inducing Adversarial Regularization

4.2.1 Version 1: Smoothness Inducing Adversarial Regularization (SIAR)

Just like how a rough sheet of metal must be sanded to achieve a smooth texture, models must also be smoothed to become more flexible than rigid. This original technique, introduced by Jiang et al., encourages the model to learn smoother decision boundaries by penalizing sharp transitions between classes. By promoting smoother boundaries, the model becomes more robust to noise and outliers, reducing overfitting and enhancing generalization. Jiang's approach is defined below in further detail as: Given a model $f(\cdot;\theta)$ and a dataset of n points $\{(x_i, y_i)\}_{i=1}^n$, where x_i represents the input data embeddings and y_i are the corresponding labels, the optimization problem for fine-tuning the model is formulated as:

$$\min_{\theta} \mathcal{L}(\theta) + \lambda_s R_s(\theta) \tag{1}$$

where $\mathcal{L}(\theta)$ is the loss function calculated as:

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} l(f(x_i;\theta), y_i) \tag{2}$$

and $R_s(\theta)$ is the smoothness-inducing regularizer:

$$R_s(\theta) = \frac{1}{n}\sum_{i=1}^{n} \max_{\|\tilde{x}_i - x_i\|_p \leq \epsilon} l(f(\tilde{x}_i;\theta), f(x_i;\theta)) \tag{3}$$

4.2.2 Version 2: Adversarial Training with Parameter Perturbation

In contrast to perturbing the input space, the adversarial training with parameter perturbation approach applies perturbations directly to the model's parameters. This method can be expressed as:

$$\tilde{\theta} = \theta + \epsilon \cdot \text{sign}(\nabla_\theta \mathcal{L}(\theta)) \tag{4}$$

where $\tilde{\theta}$ denotes the perturbed model parameters. The regular loss $\mathcal{L}(\theta)$ remains the same as in Equation 2. The adversarial loss is then computed with the perturbed parameters, leading to a combined loss:

$$\mathcal{L}_{combined} = \mathcal{L}(\theta) + \lambda_s \mathcal{L}(\tilde{\theta}) \tag{5}$$

This combined loss is then used to update the model parameters during training.

4.3 Cosine-Similarity Fine-Tuning

The cosine-similarity fine-tuning extension was applied to the similarity task. Demonstrated by Reimers and Gurevych in their work with Siamese BERT models, fine-tuning with cosine similarity adjusts model parameters based on embedding similarity (Reimers and Gurevych, 2019). Unlike conventional similarity measurements such as Euclidean and edit distance, which use geometric distance, cosine distance captures directional similarity. This makes it less sensitive to variations in vector length while still being able to capture semantic similarity (Xia et al., 2015). In our experiments, we compare cosine similarity with two types of losses: cosine embedding loss vs. mean squared error loss.

4.3.1 Version 1: With Cosine Embedding Loss

Let x_1 and x_2 be a pair of contextualized embeddings outputted from the BERT model. Let y be the list of true labels assigned to a batch of sentence pairs, indicating their similarity; in our approach, labels have been scaled to be either -1 or 1 based on the cosine embedding loss requirements. The cosine embedding loss then computes:

$$\text{loss}(x,\, y) = \begin{cases} 1 - \cos(x_1, x_2), & \text{if } y = 1 \\ \max(0, \cos(x_1, x_2) - \text{margin}, & \text{if } y = -1 \end{cases} \tag{6}$$

where $cos(x_1, x_2)$ represents the cosine similarity between the two embeddings.

4.3.2 Version 2: With Mean Squared Error (MSE) Loss

Let x_1 and x_2 be a pair of contextualized embeddings outputted from the BERT model. Let y be the list of true labels assigned to a batch of sentence pairs, indicating their similarity. In our approach, labels have been scaled to be between 0 and 1. The cosine similarity between the two embeddings is:

$$\text{logits} = \frac{x_1 \cdot x_2}{\max(\|x_1\|_2 \cdot \|x_2\|_2 \,, \epsilon)} \tag{7}$$

Mean squared error is then used to compute the loss between the cosine similarity logits v and the true y labels.

$$\ell(v, y) = L = [\ell_1 \quad \dots \quad \ell_N]^T \,, \quad \ell_n = (v_n - y_n)^2 \tag{8}$$

where N is the batch size.

4.4 Ensembling

Ensembling is a crucial enhancement in our project (and not unlike the transformation of robots into the powerful gestalt Bruticus in the *Transformers* cartoons!). In the realm of NLP transformers, ensembling combines predictions from multiple base models to generate a final, often superior prediction (Fort et al., 2020). By utilizing diverse models trained on different data and tasks, ensembling reduces overfitting and relying on a single model's predictions. We prioritize post-fine-tuning-based ensembling techniques due to hardware limitations, categorizing them into unweighted and weighted methods. This distinction highlights approaches that assign equal significance to each model, vs. those that prioritize predictions from an "expert model."

Table 1: Ensembling methods used in experiment

Unweighted Methods	Weighted Methods
Average and Majority Voting: Basic ensembling method. Aggregates outputs by averaging scores (for scoring tasks) or conducting a majority vote (for classification tasks).	**Weighted Average and Majority Voting:** Similar to its unweighted counterpart, but assigns some pre-specified weight to the "expert model" and divides the remaining weight among the "non-expert" models, amplifying the influence of the former.
Bootstrapping: Enhances ensembled predictions through resampling. Each model generates predictions, and multiple random samples with replacement are used to train a base model, capturing data variability.	**Stacking:** Combines predictions from multiple base models using a linear regression meta-model. Weights are implicitly determined during training, ensuring an unbiased approach to ensembling.

5 EXPERIMENTS

5.1 Experiment Group 1: Cosine Similarity with MSE Loss vs. with Cosine Embedding Loss

We investigated MSE Loss and Cosine Embedding Loss to discover the best version to pair with the cosine similarity extension for textual similarity tasks (the STS dataset). We also experimented with different scaling methods for the input logits and labels. We compare both extensions to our baseline model's performance (multitask fine tuning without extensions) on the SST dataset. All training experiments were performed by training and evaluating solely on the SST dataset. All test experiments were run by evaluating the model's performance on all three tasks.

5.2 Experiment Group 2: Single-task Fine-tuning with Cosine Similarity for Paraphrase Task

In this experiment, we performed single-task fine-tuning with the cosine similarity extension for the Quora dataset. Our goal was to determine whether the base multitask fine-tuned model's performance on paraphrase detection tasks was better or worse than our extension.

5.3 Experiment Group 3: Adversarial Loss Experiments with Smoothing vs. with Perturbing

For these experiments, we focused on the sentiment classification task and semantic textual similarity task datasets to evaluate the efficacy of adversarial loss functions. For the sentiment classification and semantic textual similarity task, we conducted a series of experiments contrasting the effects of the smoothness inducing adversarial regularization against direct perturbations. This allowed us to observe the impact of adversarial noise on model performance, with the intent to enhance model robustness and generalization across the different tasks.

5.4 Experiment Group 4: Ensembling

Taking the best-performing models from Experiment Groups 1 and 2, we designated one model as the "expert model" for each task: sentiment analysis, parahphrasing, and simliarity correlation. We then ran the four ensembling methods specified in our Approaches section on these models on all three tasks and the full dataset.

5.5 Data

Our models are trained on 3 datasets. For the sentiment analysis task, we used the Stanford Sentiment Treebank (SST), the Quora dataset, and the SemEval Benchmark (STS) dataset. The SST dataset consists of 11,855 movie sentence reviews for the sentiment classification task, and the Quora data consists of 400,000 question pairs with labels that are true if the sentences are paraphrases of one another for the paraphrase task. Finally, the SemEval STS Benchmark dataset facilitated the fine-tuning of semantic similarity assessment through 8,628 sentence pairs scored from 0 to 5, split into

6,041 for training, 864 for development, and 1,726 for testing. The model was pre-trained on the SST database as well as the CFIMDB dataset which contained 2,434 polarized movie reviews.

5.6 Evaluation Methods

Our evaluation methods followed the specifications of the CS224N default project. For the sentiment analysis and paraphrase detection task with categorical outcomes, we used accuracy. For the semantic textual similarity task, which involves a range of similarity scores, we used Pearson correlation as our evaluation method.

5.7 Experimental Details

Unless otherwise specified, all models were trained with a learning rate of $1 \cdot 10^{-5}$ and a batch size of 16. For the smoothness-inducing adversarial regularization term, we tested with values: $\epsilon = \{0.1, 1 \cdot 10^{-5}, 1 \cdot 10^{-5}\}$, $\lambda = \{0.1, 1, 5\}$, and $\eta = 1 \cdot 10^{-5}$.

5.8 Results

Below are two tables reporting our results on the dev and test datasets. The great disparities between the two indicate that overfitting may have occurred when running on the test set when using the weighted average and majority voting (with an expert weight of 1.00 and 0.5). While ensembling methods are used to help improve model generalization and reduce overfitting, using the expert model with the 1.00 and 0.5 weighting strategies could have led to a high level of specialization to the dev data. Consequently, while the dev set performance suggested an improved capability to handle the variability within that dataset, this did not translate well to the test set, where a different or more diverse set of examples and linguistic features might have been encountered.

Table 2: Consolidated dev results of all experimental techniques

Method	Sentiment Acc	Paraphrase Acc	Similarity Corr
Pretrain Default	0.301	0.653	-0.094
Baseline Multitask Finetuned BERT (No Extensions)	0.342	0.738	0.365
Fine-tune SST + Cosine Embedding Loss	0.208	0.375	0.427
Fine-tune SST + MSE Loss + Labels Scaled to [0, 1]	0.167	0.478	**0.855**
Finetune Paraphrase + Cosine Similarity	*	0.632	*
Param. Pert. $\epsilon = 10^{-4}$	0.505		0.313
SIAR $\epsilon = 10^{-4}$	0.501	*	0.328
Param. Pert. $\epsilon = 10^{-1}$	0.507	*	0.337
Avg + Majority Vote	0.098	0.656	0.800
Bootstrapping	0.231	0.549	-0.019
Weighted Avg + Majority Vote (Expert Weight: 0.50)	0.380	0.700	0.839
Weighted Avg + Majority Vote (Expert Weight: 0.75)	0.471	0.745	**0.855**
Weighted Avg + Majority Vote (Expert Weight: 1.00)	**0.514**	**0.745**	**0.855**
Stacking	**0.514**	0.577	0.473

Note: * indicates that the task was not tested for this method.

Table 3: Consolidated test results of weighted average and majority vote ensembling

Method	SST Accuracy	Paraphrase Acc.	STS Corr.
Weighted Avg + Majority Vote (Expert Weight 1.00)	0.210	0.749	-0.013
Weighted Avg + Majority Vote (Expert Weight 0.50)	0.225	0.708	0.051

6 ANALYSES OF DEV EXPERIMENTS

6.1 Experiment Group 1: Cosine Similarity with MSE Loss vs. with Cosine Embedding Loss

Our experiments display a significant improvement in the dev accuracy for similarity tasks when utilizing cosine similarity and mean squared error loss as opposed to cosine similarity and cosine embedding loss.

Per its requirements, cosine embedding loss takes in binary target values, where -1 and 1 represent no similarity and perfect similarity respectively. However, the paraphrase task evaluates similarity on a continuous scale between the ranges 0 to 5. In order to fit the requirements of cosine embedding loss, we mapped the true labels from the STS dataset to be either -1 or 1. Table 2 shows our implementation of cosine embedding loss improved on the baseline score from 0.231 to 0.427 for the semantic textual similarity correlation.

While the performance has improved, we believed that the binary target values required by Cosine Embedding Loss removes the complexity required to properly evaluate the similarity between embeddings. As a result, we experimented with Mean Squared Error Loss. Since Mean Squared Error (MSE) calculates the squared difference between predicted and target values without heavily restricting either to a specific range, it's more suitable for the continuous values outputted by the cosine similarity extension. However, to minimize incorrect classifications, we need to ensure both predicted and target values are on the same scale. We explored various scaling methods:

Table 4: Dev accuracy results for scaling methods with MSE loss

Method	Sentiment Acc	Paraphrase Acc	STS Correlation
No Scaling	0.440	0.640	0.323
logits * 5	0.185	**0.577**	0.815
5 * sigmoid(logits)	0.186	0.511	0.670
(logits + 1) * 2.5	0.211	0.453	0.502
Scaled labels to [-1, 1]	**0.214**	0.493	0.430
Scaled labels to [0, 1]	0.167	0.478	**0.855**

From Table 4, we see that scaling the labels to be between [0, 1] outperforms other scaling methods for the STS dataset by a large margin. This subverted our initial expectations as we believed scaling labels to be between [-1, 1] (to match the range of the logits) would outperform other methods. Upon further analysis, we see that the cosine similarity assigns a 1 for total similarity, -1 for opposite, and 0 for unrelated or orthogonal. Scaling labels to be between 0 and 1 influences the model to give dissimilarity scores of 0 instead of -1; this is the most accurate approach as we technically do not measure "opposite" sentences and orthogonality is the closest we will get to dissimilarity between a pair.

6.2 Experiment Group 2: Analysis of Multi-tasking for Paraphrase Tasks

From 2, we see that the model that was single-tasked fine-tuned on the Paraphrase (Quora) dev dataset received an accuracy of 0.632. On the other hand, the baseline multitask fine-tuned BERT model (with no extensions applied) performed much better, receiving an accuracy of 0.738. Therefore, we decided to use the baseline multitask fine-tuned BERT model in our ensemble infrastructure specifically for paraphrase tasks. A possible explanation for this disparity may be that single-task fine-tuning with the extension lowered the model's ability to analyze a broader range of paraphrase tasks where deeper connections between embeddings are required.

6.3 Experiment Group 3: Analysis of Adversarial Loss Approaches

6.3.1 Performance Evaluation

Interestingly, the parameter perturbation adversarial approach with $\epsilon = 10^{-1}$ shows the highest accuracy on the SST task with a dev accuracy of 0.507. This indicates that a higher perturbation level may facilitate better generalization for sentiment classification, possibly due to a greater exploration

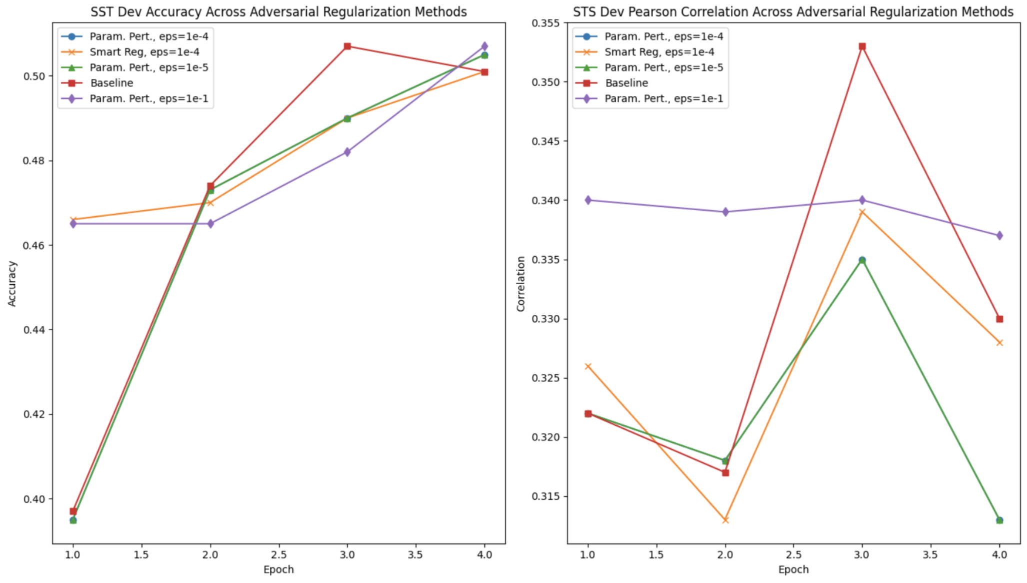

Figure 1: Adversarial loss approaches for SST and STS tasks

of the embedding space. This is followed by the parameter perturbation with $\epsilon = 10^{-4}$ with a dev accuracy of 0.505, followed by the SIAR adaptation with a dev accuracy 0.501.

In a more qualitative perspective, we speculate that the increased performance from the addition of adversarial methods for the SST task compared to the baseline could be due to the potentially beneficial effects of the perturbations: higher perturbations could lead the model to prioritize more semantically meaningful features rather than potentially misleading surface-level cues and noise. Thus, these perturbations could help the model focus on developing a deeper, more contextual understanding for the SST task. Conversely, for the STS task, due to its objective of measuring the semantic similarity between sentence pairs, adversarial perturbations, especially those with higher epsilon scores, could obscure the meaning of one or both of the sentences, thus leading to a lower semantic overlap detected by the model, potentially leading to a reduced ability to accurately predict similarity scores between sentences.

6.4 Experiment Group 4: Ensembling

Our analysis of ensemble methods across all three grammar tasks reveals some intriguing insights. First, the unweighted ensembling methods (average plus majority vote and bootstrapping) exhibit mixed performance, as seen in Table 2. Though averaging and voting gained 0.435 in performance on similarity correlation from the baseline, this boost is counteracted by drops of -0.244 and 0.082 in sentiment and paraphrase accuracy, respectively. Bootstrapping also shows less-than-stellar results, underperforming across all three tasks: -0.111, -0.189, and –0.384 for the sentiment, paraphrase, and similarity tasks, respectively. In fact, the models' worst performances were near zero: 0.098 sentiment accuracy for the average and majority vote technique, and a -0.019 similarity correlation for bootstrapping.

On the other hand, weighted ensemble methods, most likely thanks to the incorporation of expert opinions, offer more promising results. The weighted average plus majority voting technique demonstrates notable improvements, especially in the sentiment and similarity tasks. Specifically, sentiment accuracy shows an increase of approximately 0.172 when the sentiment task is done entirely by the "expert model" (weighed at 1.00, other two models weighed at 0.00), reaching 0.514 accuracy compared to the baseline's 0.342. Similarly, similarity correlation sees a substantial improvement of over double the baseline score, with an expert weight of 1.00, jumping from 0.365 to 0.855. This suggests that considering the expertise of individual models significantly enhances ensemble performance. Also, stacking, though its performance spikes aren't as high as weighted averaging and voting, still shows considerable improvement: +0.172, +0.108 for sentiment accuracy and similarity correlation above the baseline.

7 ERROR ANALYSIS ON TEST DATASET

In this section, we analyze the contrast between the dev set performance and the test set outcomes of our ensemble methods (see Table 2 vs. Table 3). When employing the weighted average and majority vote with an expert weight of 1.00, the model exhibited robust performance on the dev set but faltered significantly on the test set. This pattern suggests potential overfitting; the ensemble method's strong dev set results did not generalize to the test set. A heavy reliance on the "expert model" likely narrowed the ensemble's predictive capabilities, constraining it to the idiosyncrasies of the dev set and impairing its performance on the diverse examples of the test set.

Our hypothesis is that the ensemble's overfitting was caused by an overemphasis on the dev set characteristics, reinforced by the "expert model" weight of 1.00, leading to a finely tuned performance on the dev dataset at the expense of broader applicability. The incremental improvements observed with an expert weight of 0.50 on the SST and STS tasks—despite overall lower scores lend credence to our assumption. This shift in weights, while still not optimal, reduced the over-specialization to the dev data, thereby confirming our hypothesis and underscoring the need for a balanced approach that better captures the variance across datasets.

Additionally, our approach to fine-tuning could have contributed to the error. We applied single-task fine-tuning for SST and STS, yet multitask fine-tuning for paraphrase detection. Furthermore, our final results from the test dataset 3 show how well the paraphrase dataset did compared to the other two. This proves that, contrary to our initial assumptions, our fine-tuning approach for the ensembling techniques did not promote generalization to unseen tasks nor facilitate transfer learning between different task types, highlighting the need for a re-evaluation of fine-tuning strategies in pursuit of more generalizable models.

8 CONCLUSION

Our study explored various fine-tuning techniques and architectures to optimize BERT's performance across sentiment analysis, paraphrase identification, and textual similarity tasks. Notably, we delved into single-task vs. multitask fine-tuning, and we also identified optimal ensemble techniques for single-task fine-tuned models.

For single-task vs. multitask fine-tuning, we observed that cosine similarity provided significant improvements in textual similarity tasks, regardless of fine-tuning approach used. While adversarial loss does enhance sentiment accuracy, its impact on similarity tasks was inconclusive, suggesting a nuanced balance between robustness and semantic fidelity.

While there were significant discrepancies between the dev and test results, our research still underscored potential benefits of ensembling fine-tuned models to enhance performance. Our experiments revealed that despite the overarching issue of overfitting, the weighted average plus majority voting approach, particularly when heavily weighting the "expert model," did show promise under controlled conditions. This finding suggests that while the methodology did not perform as expected universally, there remains value in exploring domain-specific expertise within model ensembling, albeit with a more nuanced approach to avoid overfitting and ensure better generalizability.

Our findings not only contribute to the discourse on effective fine-tuning and ensemble methods in natural language processing but also, it prompts reflection on the dynamic between task specialization and combination. Looking ahead, avenues for future exploration include extensions of adversarial loss functions, such as the Bregman proximal point approximation, to further enhance model robustness and generalization. Additionally, we propose exploring multitask fine-tuning with weighted averaging of development results, aiming to optimize model performance across multiple tasks simultaneously.

In conclusion, our research highlights the significance of domain expertise and task-specific fine-tuning for optimizing minBERT's performance. By leveraging diverse fine-tuning strategies and ensemble techniques, we advance our understanding of effective approaches in NLP, exploring opportunities for more robust and flexible language models.

References

Bian, Yijun Wang, Yijun Yao, Yaqiang Chen, Huanhuan. (2019). Ensemble Pruning Based on Objection Maximization With a General Distributed Framework. IEEE Transactions on Neural Networks and Learning Systems. PP. 1-9. 10.1109/TNNLS.2019.2945116.

Devlin, Jacob et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. North American Chapter of the Association for Computational Linguistics (2019).

Fort, Stanislav, et al. Deep Ensembles: A Loss Landscape Perspective. *arXiv preprint arXiv:1912.02757*.

Mienye, Domor Sun, Yanxia. (2022). A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects. IEEE Access. PP. 1-1. 10.1109/ACCESS.2022.3207287.

Kuhn, Max, and Kjell Johnson. (2018). *Applied predictive modeling*. Springer.

Reimers, Nils, and Iryna Gurevych. Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.

Vaswani, Ashish, et al. Attention Is All You Need. *Advances in Neural Information Processing Systems*.

Xia, Peipei Zhang, Li Li, Fanzhang. (2015). Learning Similarity with Cosine Similarity Ensemble. Information Sciences. 307. 10.1016/j.ins.2015.02.024.

Zhang, Yu Yang, Qiang. (2018). An overview of multi-task learning. National Science Review. 5. 30-43. 10.1093/nsr/nwx105.