SLOTH: Semantic Learning Optimization and Tuning Heuristics for Enhanced NLP with minBERT

Stanford CS224N Default Project

Phillip Miao
Department of Computer Science
Stanford University
pmiao@stanford.edu

Cici Hou
Department of Computer Science
Stanford University
xhou@stanford.edu

Abstract

Advancements in Natural Language Processing (NLP) through models like BERT have significantly propelled the field, yet challenges persist in tasks such as sentiment analysis, paraphrase detection, and semantic textual similarity due to the nuanced nature of human language. This paper presents an innovative approach to these tasks by enhancing the BERT model with Multiple Negatives Ranking Loss (MNRL) and regularized optimization techniques, specifically aimed at improving semantic understanding and model generalization. We introduced fine-tuning strategies that leverage task-specific head variations and pre-concatenation of sentence pairs, offering a deeper contextual analysis. Our experimental results exhibit notable enhancements across all tasks, with our methods achieving leading scores on benchmark leaderboards. The implementation of MNRL demonstrated a refined ability to discriminate between closely related texts, whereas the regularization methods effectively mitigated the overfitting problem prevalent in deep learning models. This study highlights the critical aspects of model tuning and loss function selection in achieving superior NLP task performance.

1 Key Information to include

- Mentor: Johnny Chang
- External Collaborators (if you have any): No
- Sharing project: No

2 Introduction

Natural Language Processing (NLP) has witnessed remarkable progress with the advent of advanced models, yet certain tasks such as sentiment analysis, paraphrase detection, and semantic textual similarity remain challenging due to the intricacies of human language. These tasks are crucial for various applications including sentiment prediction in reviews [1], identification of duplicate questions in forums [2], and accurate understanding of sentence meaning, which can enhance machine-human interactions and information retrieval systems [3].

The difficulty in these tasks arises from the complexity of natural language, which includes nuances such as context, sarcasm, idiomatic expressions, and subtle semantic differences [4]. Current state-of-the-art models, such as BERT (Bidirectional Encoder Representations from Transformers) [5], have significantly improved performance by leveraging deep bidirectional context understanding. However, they still face limitations like overfitting during fine-tuning, inability to capture all semantic nuances, and computational inefficiency [6].

Our project aims to address these challenges by enhancing BERT through several approaches. We investigate the use of Multiple Negatives Ranking Loss (MNRL) to better distinguish between semantically similar and dissimilar sentence pairs, thereby improving paraphrase detection [7]. Additionally, we implement fine-tuning with regularized optimization techniques to combat overfitting and enhance generalization across various tasks [8]. Our methodology includes experimenting with task-specific head variations and pre-concatenation of sentence pairs to leverage contextual relationships more effectively.

Our results demonstrate significant improvements in model performance across all three tasks, achieving the **highest score on both the Dev and Test leaderboards** (by 9 pm on June 8th). In the following sections, we delve into the related work that informed our approach, the detailed methodology employed, and a thorough evaluation of our experimental results. The findings highlight the importance of pre-concatenating sentence pairs before inputting them into the BERT model and fine-tuning task-specific head hyperparameters. Additionally, they underscore the value of regularized fine-tuning in enhancing the optimization process.

3 Related Work

3.1 Background of BERT

Before the advent of BERT (Bidirectional Encoder Representations from Transformers) [5], various language models significantly contributed to the development of natural language processing (NLP). Early models like Word2Vec [9] and GloVe [10] laid the foundation for word embeddings by capturing semantic relationships between words through context. However, these models had limitations, as they produced static, context-independent embeddings for words.

Later, context-dependent models emerged, addressing the shortcomings of static embeddings. The ElMo (Embeddings from Language Models) model [11] introduced contextualized word embeddings by using a bidirectional LSTM to process text, capturing context from both directions. This approach significantly improved the performance of various NLP tasks. Similarly, the GPT (Generative Pre-trained Transformer) model [12] utilized a unidirectional transformer architecture, showing that pre-training on a large corpus followed by fine-tuning could yield state-of-the-art results in several NLP benchmarks.

3.2 Bidirectional Encoder Representations from Transformers (BERT)

BERT, introduced by Devlin et al. (2019), marked a significant breakthrough in NLP by implementing a bidirectional transformer architecture [5]. Unlike previous models, BERT was designed to pre-train deep bidirectional representations by jointly conditioning on both left and right contexts in all layers. This approach allowed BERT to achieve a more profound understanding of language context and nuance.

BERT's architecture consists of several transformer layers [13] that enable it to capture intricate patterns in the text. The model's training process includes two key unsupervised tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). MLM involves randomly masking words in a sentence and training the model to predict them, thereby allowing it to learn context-dependent word representations. NSP helps the model understand the relationship between sentences by predicting whether a given sentence follows another in the corpus. In our project, we utilize pre-trained BERT, adding task-specific heads and fine-tuning them to serve as our project's baseline.

3.3 Multiple Negatives Ranking Loss

Following BERT's success, numerous methods have been developed to further enhance language understanding capabilities. One such method is the Multiple Negatives Ranking Loss (MNRL), which was introduced to improve the efficiency and scalability of training ranking models [7]. MNRL operates by treating all other examples in the training batch as negative samples for a given positive example. This mechanism aims to minimize the distance between similar sentences while maximizing the distance between dissimilar pairs.

MNRL is particularly well-suited for our paraphrase detection task, as the objective aligns with distinguishing between semantically similar and dissimilar sentence pairs. We experimented with MNRL as an additional fine-tuning step. Our goal was to assess whether this approach could enhance the model's ability to discern subtle semantic differences between sentences.

3.4 Fine-Tuning with Regularized Optimization

Fine-tuning pre-trained language models often faces the challenge of overfitting, which can hinder the model's ability to generalize to unseen data. To address this issue, Jiang et al. introduced the SMART method, which stands for Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization [8]. SMART employs two primary strategies: Smoothness-inducing regularization and Bregman proximal point optimization, both designed to ensure robust and efficient fine-tuning.

In our work, we applied the SMART methodology across three tasks: paraphrase detection, sentiment analysis, and semantic textual similarity. For each of these tasks, the principles of regularized optimization provided by SMART were instrumental in achieving robust fine-tuning. The application of SMART in our experiments demonstrated its versatility and effectiveness in fine-tuning pre-trained language models across diverse NLP tasks.

4 Approach

4.1 Baseline

We implemented the minBERT model, a minimalist implementation of the BERT model [5], and added our classifier and the Adam optimizer. For the baseline, we finetuned the last linear layer and the full model on the SST, Quora, and SemEval datasets to perform all three tasks and evaluated our results. Specifically, we used cross entropy loss, binary cross entropy loss, and mean square loss as the loss function for sentiment analysis, paraphrase detection and sementic textual similarity respectively. The specific architecture for the model is shown in Figure 1(a) and 1(b).

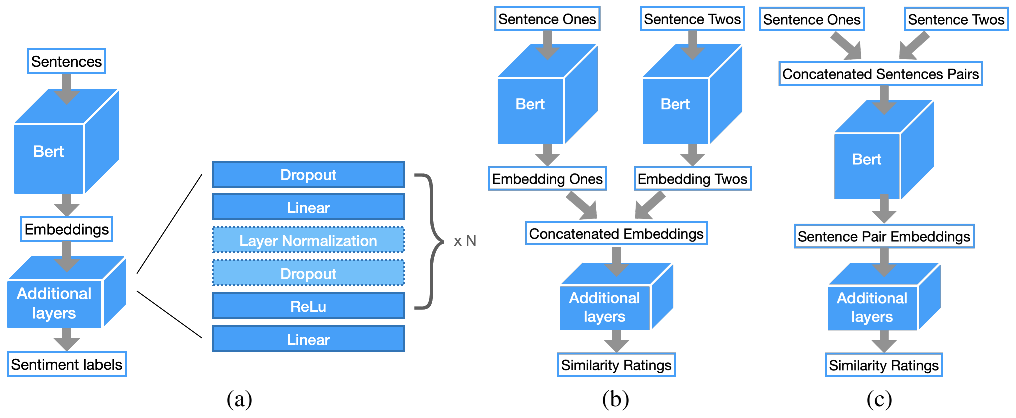


Figure 1: The architectures of our baseline models and pre-concatenation models. (a) is our baseline model for sentiment analysis. (b) is our baseline model for paraphrase detection and semantic textual similarity. (c) is the pre-concatenation model for paraphrase detection and semantic textual similarity.

4.2 Sentence Pair Pre-Concatenation

We experimented with the pre-concatenation of sentence pairs before inputting them into our BERT model, generating one unified embedding for each pair as shown in Figure 1(c). This method involves concatenating two sentences into a single input sequence by passing them through the tokenizer and then feeding this concatenated sequence into the BERT model. By doing this, we aim to leverage BERT's capability to understand and encode the relationship between the two sentences more effectively.

4.3 Multiple Negatives Ranking Loss (MNRL) Learning

The MNRL Learning method aims to minimize the distance between similar sentences while maximizing the distance between dissimilar pairs [7]. Please refer to the Appendix Methodology Details section for the specific method to calculate per-batch MNRL.

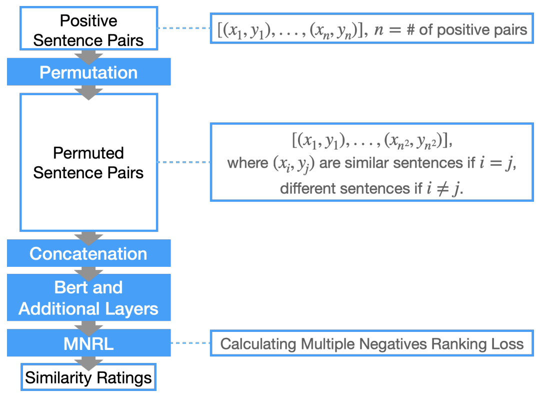

Positive Sentence Pairs
$[(x_1, y_1), \ldots, (x_n, y_n)]$, n = # of positive pairs
Permutation
Permuted Sentence Pairs
$[(x_1, y_1), \ldots, (x_{n^2}, y_{n^2})]$, where (x_i, y_j) are similar sentences if $i = j$, different sentences if $i \neq j$.
Concatenation
Bert and Additional Layers
MNRL
Calculating Multiple Negatives Ranking Loss
Similarity Ratings

Figure 2: MNRL Learning Model Structure

In our study, we employed the MNRL Learning method as an additional fine-tuning step for the paraphrase detection task, as illustrated in Figure 2. We conducted experiments to fine-tune the model using MNRL on two types of datasets: those containing only positive sentence pairs, and those comprising both positive and negative sentence pairs. This approach allowed us to assess the effectiveness of MNRL in enhancing the model's capability to accurately identify paraphrases.

4.4 Fine-Tuning with Regularized Optimization

Aggressive fine-tuning often leads to overfitting, causing the model to fail to generalize to unseen data. To address this issue, we adopt two approaches proposed by Jiang et al. across all three tasks: (1) Smoothness-inducing regularization, which effectively manages model complexity, and (2) Bregman proximal point optimization, a trust-region method that prevents aggressive updates [8]. Specific loss functions and optimization equations are listed in Appendix A.1.2 and A.1.3.

4.5 Task-Specific Head Variation

Finally, we experimented with various variations of the task-specific head to examine the combination that contributed to the best performance. The parameters we tuned included: the number of linear layers, the dropout rate, and with or without layer normalization.

5 Experiments

In this section, we present a detailed account of the experiments conducted to evaluate the performance of our proposed methods across three NLP tasks. Our goal was to assess the effectiveness of various strategies, including sentence pair pre-concatenation, Multiple Negatives Ranking Loss (MNRL), Regularized Optimization, and Task-Specific Head Variation in improving the performance of BERT-based models.

5.1 Data

Table 1 shows the datasets we are using with regard to their tasks.

Task	**Dataset**	**Train #**	**Dev #**	**Test #**
Sentiment Analysis	SST	8,544	1,101	2,210
	CFIMDB	1,701	245	488
Paraphrase Detection	Quora	283,010	40,429	80,859
Semantic Textual Similarity	SemEval STS Benchmark	6,040	863	1,725

Table 1: Datasets for Different Tasks

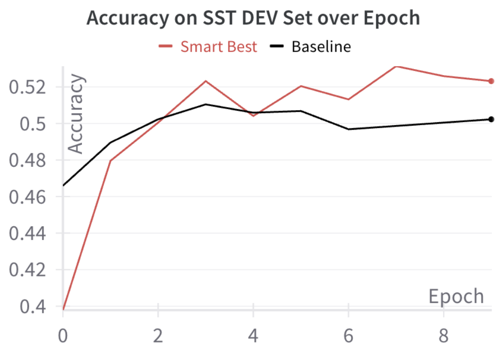
5.2 Evaluation method and Experimental Details

For sentimental analysis and paraphrase detection, we will use accuracy to evaluate performance. For semantic textual similarity, we use, as in the original SemEval paper [14], Pearson correlation of the true similarity values against the predicted similarity values.

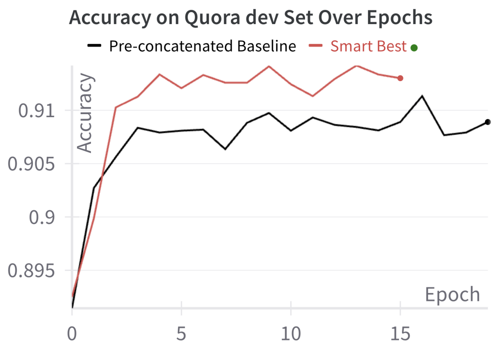
For most of our experiment unless specified, we use learning rate equals to 10^{-5} and a batch size equals to 64 for semantic textual similarity and sentimental analysis and 16 for paraphrase detection due to memory limitations. The maximum number of epochs was set to 20. A linear learning rate decay schedule with warm-up of 0.1 was used. To avoid gradient exploding, we clipped the gradient norm within 1. For regularized optimization, we set the perturbation size $\epsilon = 10^{-5}$ and $\sigma = 10^{-5}$. We set $\mu = 1$ and the learning rate η for the gradient desent step for solving Equation 4 is set to 10^{-3}.

5.3 Sentence Pair Pre-Concatenation

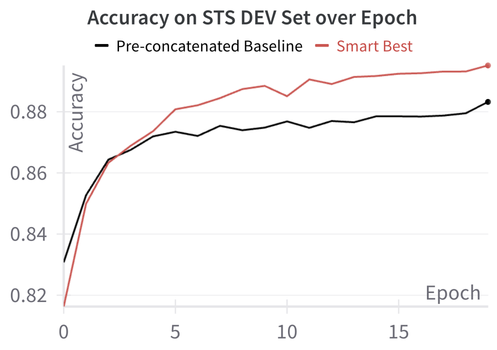
During our experiments, we found that pre-concatenating sentence pairs and generating a single embedding for each pair resulted in significantly improved performance compared to treating the sentences independently. The pre-concatenation method performs consistently better than the original method on paraphrase detection (from 78.61% to 91.13%) and semantic textual similarity tasks (from 43.17% to 88.32%). The specific accuracy curve comparison can be found in Appendix A.2.2. This improvement is likely due to the enhanced contextual information that BERT can capture when the sentences are processed together. This is the most important improvement that we see in our experiments.



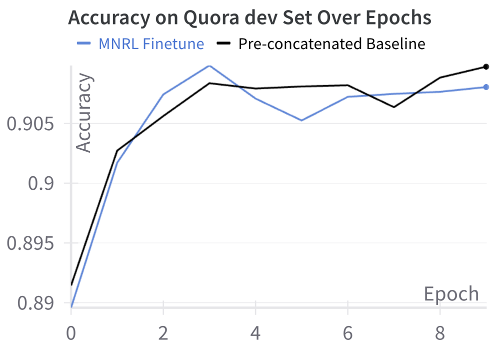
(a)



(b)



(c)



(d)

Figure 3: The accuracy curve on the dev dataset over training epochs. Figure (a), (b), (c) compare the Pre-concatenated Baseline with the best-performing model with Regularized Optimization on each of the three tasks. Figure (d) compares MNRL Finetuning model with the Pre-concatenated Baseline.

5.4 Multiple Negatives Ranking Loss Experiments

We implemented Multiple Negative Ranking Loss (MNRL) for the paraphrase detection task and used it as the primary loss function. Since MNRL is designed to be used with positive pairs, we filtered the dataset to include only positive pairs. For each batch, we generated all possible permutations between sentence one and twos as training samples. With pre-concatenation of sentence pairs, we receive an accuracy of 59.86% on the dev set. The accuracies are lower than the baseline accuracy. We believe the main issue is that we trained only on positive pairs, a biased portion of data samples. We concluded that MNRL should be employed as an additional fine-tuning loss alongside the original binary cross-entropy loss. With this approach, we see a faster convergence to a higher accuracy on dev set as shown in Figure 3(d) although the improvement is very subtle.

5.5 Fine-Tuning with Regularized Optimization

We observed numerous instances of overfitting in our model across all three tasks—evidenced by a substantial gap between dev and training accuracy, as well as a decline in dev accuracy towards the end of training—we utilized regularized optimization techniques. Therefore, we implement both Smoothness-Inducing Adversarial Regularization (SIAR) and Bregman Proximal Point Optimization (BPPO) as a way to regularize our training process. The impact of these regularization methods is substantial, as illustrated in Figure 3(a), 3(b), 3(c). These techniques contributed to the model's convergence towards a more optimal solution by effectively reducing overfitting. The implementation of SIAR ensures that the model maintains smoothness in its predictions, reducing sensitivity to minor fluctuations in the training data. Meanwhile, BPPO aids in stabilizing the optimization process, ensuring that the model does not deviate too far from previous iterates, thereby enhancing generalization performance.

5.6 Task-Specific Head Variation

After implementing all the previous extensions, we conducted a grid search to determine the optimal combination of hyperparameters for our task-specific head. We initially observed that tuning the entire model yields significantly better results than tuning only the classification head; therefore, we performed the grid search exclusively on the fully-tuned model. The parameters we adjusted included the number of linear layers, the dropout rate, and the presence or absence of layer normalization. The results are displayed in Table 3 for all three tasks. Notably, we did not complete the full model grid search for paraphrase detection due to the lengthy runtime of these experiments; instead, we focused on the two hyperparameters that achieved the best results for SST and STS. We observed an increase in performance after selecting the optimal hyperparameter combination for the model head.

# of layers	LayerNorm	Dropout	SST	Quora	STS
0	No	0.1	**53.95**	91.42	89.43
		0.4	53.68	-	89.44
2	No	0.1	52.50	-	89.35
		0.4	51.32	-	89.08
2	Yes	0.1	52.41	**91.47**	**89.59**
		0.4	51.95	-	89.25

Table 2: Grid search results to determine the optimal combination of hyperparameters for our task-specific head for all three tasks.

5.7 Test Results

After implementing various models and tuning hyperparameters, we submitted our best-performing models to the test leaderboard. Interestingly, only the best-performing model on the Quora dataset in the dev set achieved the best performance on the test set. For both SST and STS, the highest test leaderboard scores were obtained with the second-best-performing models from the dev set. This discrepancy is primarily due to overfitting. The second-best-performing model utilized a dropout rate of 0.4 instead of 0.1, making it less prone to overfitting. As a result, our test accuracy actually increased compared to the dev accuracy. As of the time of writing this report (9 pm on June 8th), we have achieved the **highest score on both the Dev and Test leaderboards** with the both the overall score equals to 80.1%.

Task	# of layers	LayerNorm	Dropout	SMART	MNRL	Dev Acc	Test Acc
SST	0	No	0.4	Yes	-	53.7	53.9
Quora	2	Yes	0.1	Yes	Yes	91.5	91.5
STS	0	No	0.4	Yes	-	89.4	89.5
Overall	-	-	-	-	-	80.1	**80.1**

Table 3: Best Dev and Test leaderboard results for all three tasks.

6 Analysis

6.1 Multiple Negatives Ranking Loss

From our predictions on the DEV dataset, we observed that the MNRL method performs exceptionally well on positive sentence pairs but exhibits instability on negative sentence pairs. For instance, the model accurately predicted similarity for the complex and structurally diverse pair, "Why is Narendra Modi promoting Reliance Jio?" and "Why did Narendra Modi allow Reliance to publish his photo on their Jio ad?" This demonstrates the model's ability to handle intricate sentence structures effectively. Conversely, for dissimilar pairs that could be easily distinguished by human, "I bought a 35 day old pug puppy. I know it is really too young for it to be away from his mother. How do I take care of him and make him healthy?" and "Can I give Himalaya Digyton to a 27 day old pug puppy?", or "How much money do you need to start a new life?" and "How do I disappear and start a new life?", the model incorrectly assigned positive ratings, indicating similarities between pairs. This misclassification suggests that the model, predominantly trained on positive cases, lacks sufficient experience with negative cases. Consequently, while the model achieves high accuracy in identifying paraphrases, it struggles to accurately distinguish non-paraphrase pairs. This imbalance in training data likely contributes to its reduced performance in detecting dissimilar sentences.

6.2 Best Performing Model Error Analysis

6.2.1 Sentimental Analysis

The misclassified pairs reveal specific challenges in sentiment analysis, particularly with sarcasm, humor, and contextual understanding. Here are some illustrative examples:

- Example 1: "movie fans, get ready to take off . . . the other direction." misclassified as 4 instead of 1.
- Example 2: "the words, 'frankly, my dear, i don't give a damn,' have never been more appropriate." misclassified as 4 instead of 1.

Sentences with sarcasm or humor, as seen in Example 1, pose significant difficulties for the model. Example 2 highlights the model's struggle with nuanced context, particularly with phrases that have cultural or historical significance. These misclassifications suggest that the model lacks a comprehensive understanding of context and the subtleties of sarcasm, leading to errors in sentiment prediction.

6.2.2 Paraphrase Detection

We examined the misclassified pairs and provided some examples as follows:

False Positives (Predicted 1, True 0):

- Example 1: "what is the average salary of an architect?" vs. "what is the average salary of a business architect?"
- Example 2: "what do successful people do?" vs. "what are some little things that successful people do?"

False Negatives (Predicted 0, True 1):

- Example 1: "how can i interest my thirteen year old neighbour to get into programming?" vs. "how can i use minecraft to get my 13 year old interested in programming?"
- Example 2: "where can you go to look up a license plate and owner of a car without any charge?" vs. "how do you look up license plate numbers?"

We can generalize some patterns in our errors. For example, sentences with slight differences in context or specific details for example "architect" vs. "business architect" seem to cause confusion and make the model believe they are paraphrases. Besides, minor changes in phrasing or added details can lead to misclassification, for example, "how do you look up license plate numbers ?" vs. "where can you go to look up a license plate and owner of a car without any charge ?"). The potential improvements can be introducing more training data with similar contextual variations to help the model learn to distinguish subtle differences and use more advanced models that can better capture the nuances in language.

6.2.3 Semantic Textual Similarity Analysis

The misclassified pairs in the semantic textual similarity task highlight specific challenges in accurately assessing similarity between sentences. Below are some selected examples and a brief analysis:

- Example 1: "world stocks rise on hopes fed to keep stimulus" and "fed expected to maintain stimulus" with a predicted similarity score of 3.6140003204345703, but a true score of 1.0.
- Example 2: "work into it slowly." and "it seems to work." with a predicted similarity score of 3.558607578277588, but a true score of 0.0.
- Example 3: "it's also a matter of taste." and "it's definitely just a matter of preference." with a predicted similarity score of 2.084804058074951, but a true score of 5.0.

Notably, the model struggles with understanding and differentiating the context in which similar phrases are used, as evidenced by the high similarity scores for sentences with common words but differing meanings, as shown in Example 1 and 2. Furthermore, the model demonstrates a weakness in recognizing synonymous phrases, leading to low similarity scores for sentences that convey the same sentiment, as shown in Example 3.

7 Conclusion

In this project, we focused on improving the performance of BERT-based models for three critical NLP tasks: paraphrase detection, sentiment analysis, and semantic textual similarity. Our approach incorporated advanced techniques including Multiple Negatives Ranking Loss (MNRL), fine-tuning with regularized optimization, and pre-concatenation of sentence pairs. These methods aimed to enhance the model's ability to distinguish between semantically similar and dissimilar sentences, mitigate overfitting, and leverage contextual relationships more effectively.

Our experiments demonstrated significant improvements in model performance across all three tasks. The use of pre-concatenation notably boosted the accuracy of paraphrase detection and semantic textual similarity, highlighting the importance of capturing contextual relationships between sentence pairs. Fine-tuning with regularized optimization, specifically using Smoothness-Inducing Adversarial Regularization (SIAR) and Bregman Proximal Point Optimization (BPPO), played a crucial role in addressing overfitting challenges and stabilizing the optimization process. However, while MNRL provided valuable insights, its effectiveness was limited when used as the sole loss function due to insufficient exposure to negative examples.

Despite these achievements, our work faced several limitations, such as the model's difficulty in handling nuanced language features like sarcasm, idiomatic expressions, and contextual subtleties. The computational demands of fine-tuning and the complexity of hyperparameter optimization also posed constraints. Future work could focus on incorporating more diverse training data with a broader range of contextual variations and exploring advanced language models with enhanced contextual understanding capabilities. Overall, this project highlighted the potential of combining innovative techniques with robust optimization strategies to advance the state of NLP models, providing a strong foundation for future research aimed at achieving greater accuracy and generalization in natural language understanding tasks.

8 Ethics Statement

Our project, aimed at enhancing BERT-based models for NLP tasks, poses several ethical challenges and potential societal risks. One major concern is the propagation of biases present in the training data, which can lead to discriminatory outcomes in applications such as hiring and loan approvals. To mitigate this, it is crucial to ensure diverse and representative training datasets and implement bias detection and correction mechanisms throughout the model development process. Additionally, the potential misuse of NLP models for generating harmful content, such as fake news and misinformation, necessitates the development of robust content verification systems and user education on responsible AI use.

Privacy concerns associated with large-scale data collection for training NLP models also need to be addressed. Sensitive personal information can be inadvertently included in training datasets, leading to potential privacy breaches. Implementing stringent data anonymization techniques, adhering to data protection regulations like GDPR, and maintaining transparent data governance practices can help mitigate these risks. By proactively addressing these ethical challenges, we can promote the responsible development and deployment of NLP technologies, ensuring they benefit society while minimizing potential harms.

References

[1] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard, editors, *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[2] Yun Zhang, David Lo, Xin Xia, and Jian-Ling Sun. Multi-factor duplicate question detection in stack overflow. *Journal of Computer Science and Technology*, 30:981–997, 2015.

[3] Huimin Chen, Maosong Sun, Cunchao Tu, Yankai Lin, and Zhiyuan Liu. Neural sentiment classification with user and product attention. In *Conference on Empirical Methods in Natural Language Processing*, 2016.

[4] Bo Pang, Lillian Lee, et al. Opinion mining and sentiment analysis. *Foundations and Trends® in information retrieval*, 2(1–2):1–135, 2008.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[6] Xue Ying. An overview of overfitting and its solutions. In *Journal of physics: Conference series*, volume 1168, page 022022. IOP Publishing, 2019.

[7] Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun hsuan Sung, Laszlo Lukacs, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. Efficient natural language response suggestion for smart reply, 2017.

[8] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. SMART: robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *CoRR*, abs/1911.03437, 2019.

[9] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[10] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[11] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.

[12] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. *Technical report*, OpenAI, 2018.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[14] Eneko Agirre, Daniel Matthew Cer, Mona T. Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. *sem 2013 shared task: Semantic textual similarity. In *International Workshop on Semantic Evaluation*, 2013.

A Appendix (optional)

A.1 Methodology Details

A.1.1 Multiple Negatives Ranking Loss

The MNRL function operates on training data consisting of sets of K sentence pairs $[(a_1, b_1), \dots, (a_n, b_n)]$, where each pair (a_i, b_i) is labeled as similar, and all other pairs (a_i, b_j) with $i \neq j$ are labeled as dissimilar. The MNRL minimizes the distance between similar sentence pairs (a_i, b_i) while maximizing the distance between dissimilar pairs (a_i, b_j). Specifically, the training objective is to minimize the approximated mean negative log probability of the data. For a single batch, this is calculated as shown in Equation 1, where θ represents the word embeddings and neural network parameters used to calculate S, a scoring function.

$$\begin{aligned} \mathcal{J}(x, y, \theta) &= -\frac{1}{K} \sum_{i=1}^{K} \log P_{\text{approx}}(y_i | x_i) \\ &= -\frac{1}{K} \sum_{i=1}^{K} \left[S(x_i, y_i) - \log \sum_{j=1}^{K} e^{S(x_i, y_j)} \right] \end{aligned} \tag{1}$$

A.1.2 Smoothness-Inducing Adversarial Regularization

Given the model $f(\cdot; \theta)$ and n data points of the target task denoted by $\{(x_i, y_i)\}_{i=1}^{n}$ where x_i represents the embeddings of input sentences from the language model's first embedding layer and y_i are the associated labels, Jiang et al.'s method solves the following fine-tuning optimization problem:

$$\min_{\theta} \mathcal{L}(\theta) + \lambda_s \mathcal{R}_s(\theta) \tag{2}$$

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

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i; \theta), y_i), \tag{3}$$

and $\ell(\cdot, \cdot)$ is the task-dependent loss function: cross-entropy for sentiment analysis and paraphrase detection, and mean squared error for semantic textual similarity. The term $\lambda_s > 0$ is a tuning parameter, and $\mathcal{R}_s(\theta)$ is the smoothness-inducing regularizer defined as:

$$\mathcal{R}_s(\theta) = \frac{1}{n} \sum_{i=1}^{n} \max_{\|\tilde{x}_i - x_i\|_p \leq \epsilon} \ell_s(f(\tilde{x}_i; \theta), f(x_i; \theta)), \tag{4}$$

where $\epsilon > 0$ is another tuning parameter. For sentiment analysis and paraphrase detection, $f(\cdot; \theta)$ outputs a probability simplex and ℓ_s is chosen as the symmetrized KL-divergence:

$$\ell_s(P, Q) = D_{KL}(P \| Q) + D_{KL}(Q \| P), \tag{5}$$

while for semantic textual similarity, $f(\cdot; \theta)$ outputs a scalar and ℓ_s is chosen as the squared loss:

$$\ell_s(p, q) = (p - q)^2. \tag{6}$$

A.1.3 Bregman Proximal Point Optimization

Jiang et al. also propose a Bregman proximal point optimization method to solve Equation 2. This method applies a strong penalty at each iteration to prevent aggressive updates. Using a pre-trained model as the initialization denoted by $f(\cdot;\theta_0)$, the vanilla Bregman proximal point (VBPP) method updates as follows at the $(t+1)$-th iteration:

$$\theta_{t+1} = \arg\min_{\theta} \mathcal{F}(\theta) + \mu D_{\text{Breg}}(\theta, \theta_t) \tag{7}$$

where $\mu > 0$ is a tuning parameter and $D_{\text{Breg}}(\cdot,\cdot)$ is the Bregman divergence defined as:

$$D_{\text{Breg}}(\theta, \theta_t) = \ell_s(f(\tilde{x}_i;\theta), f(x_i;\theta_t)), \tag{8}$$

with ℓ_s defined in Equation 5 and Equation 6.

If you wish, you can include an appendix, which should be part of the main PDF, and does not count towards the 6-8 page limit. Appendices can be useful to supply extra details, examples, figures, results, visualizations, etc. that you couldn't fit into the main paper. However, your grader *does not* have to read your appendix, and you should assume that you will be graded based on the content of the main part of your paper only.

A.2 Additional Results and Plots

A.2.1 Baseline Task-Specific Head Variation Table

Number of layers / dropout values		0.1	0.4
Last Linear Layer	0	0.471	0.466
	2	0.475	0.468
	2 with normalization	0.475	0.466
Full Model	0	0.510	**0.519**
	2	0.512	0.507
	2 with normalization	0.513	0.507

Table 4: Performance metrics for Sentiment Analysis

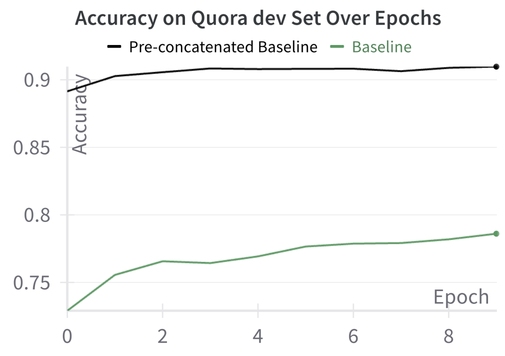
Number of layers / dropout values		0.1	0.4
Last Linear Layer	0	0.710	0.706
	2	0.771	0.749
	2 with normalization	**0.786**	0.765

Table 5: Performance metrics for Paraphrase Detection

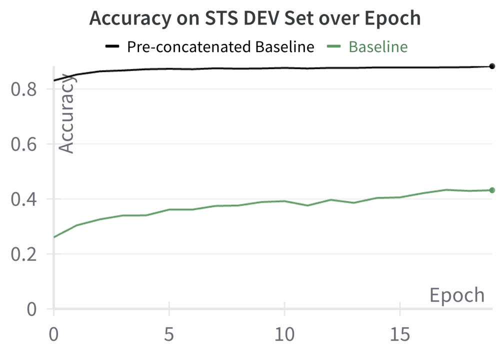
Number of layers / dropout values		0.1	0.4
Last Linear Layer	0	0.346	0.343
	2	0.427	0.383
	2 with normalization	0.**433**	0.390
Full Model	0	0.378	0.383
	2	0.377	0.370
	2 with normalization	0.404	0.377

Table 6: Performance metrics for Semantic Textual Similarity

A.2.2 Pre-concatenation vs. Baseline Accuracy Curve



(a)



(b)

Figure 4: The accuracy curve on the dev dataset over training epochs, comparing the baseline with pre-concatenated baslin.