BEES: Bi-Encoder Ensembles with Simple Contrastive Learning and Smoothness Induced Adversarial Regularization

Stanford CS224N Default Project

Nithish Kaviyan Dhayananda Ganesh
Stanford Center for Professional Development
Stanford University
nithish@stanford.edu

Abstract

This project aims to improve the performance of vanilla BERT on the following downstream tasks: sentiment analysis, paraphrase detection and similarity detection. Bi-Encoder architecture is used to overcome the computational overhead of cross-encoders especially during inference, for tasks that involve comparing two sentences. A Siamese network with pooling layer is built on top of BERT and trained on single-task and multi-task settings. This architecture improved the performance of baseline for all the downstream tasks. It is also empirically shown that introducing gradient surgery improves the performance of multi-task models. Further pre-training of the BERT model with contrastive learning and smoothness induced adversarial regularization improved the generalization of the model. An ensemble of the single and multi-task models pushed the overall performance to **79%** on the development set and obtained a test set performance of **78.6%**.

1 Key Information to include

- Mentor: Arvind Mahankali ; External Collaborators, Sharing project : No

2 Introduction

BERT (Devlin et al. (2019)) transformed the LLM suite of models by setting new state-of-the-art fine-tuning performance on different classification and similarity detection tasks. However, the BERT model which uses a cross-encoder architecture (both sentences are passed together to the BERT encoder for similarity detection tasks) suffers from very high computational overhead during inference, for similarity detection and other tasks that involve comparison of a pair of sentences. For a sample of *n* sentences, the output embeddings from the BERT model has to be estimated for *n(n-1)/2* sentences which involves each possible combination of the sentences. For example, when $n = 10000$, the task requires 499,500 outputs from the BERT model which makes the cross-encoder architecture computationally challenging for tasks that involve comparing two sentences.

The objective of this project is to improve the downstream performance of minBERT (also referred as BERT in this report) on the following tasks: sentiment classification, paraphrase detection and similarity detection. A bi-encoder architecture based on Reimers and Gurevych (2019) is applied to mitigate the above mentioned shortcomings of cross-encoders. The Siamese network with pooling layer improves the performance of pre-trained BERT on all three downstream tasks. This architecture also offers a computational benefit where each sentence can be passed once through the BERT network to obtain the contextual embeddings i.e. a sample of *n* sentences would require only *n* passes through the BERT encoder. Contrastive learning framework Gao et al. (2021), which improves the sentence embeddings of similarity tasks, was experimented by pre-training the BERT network with the training

input sentences of the downstream tasks and it was observed that the task performances improved further. Since fine-tuning on smaller datasets are prone to overfitting, smoothness induced adversarial regularization (Haoming et al. (2019)) was applied to improve the generalization performance of the fine-tuned models. Multi-task learning are known to improve the performance of the tasks by facilitating parameter sharing among different tasks, allowing them to learn from one another. In the multi-task setting, sampling strategies such as min-size Round Robin (RR), hybrid-sampling (HS) and Annealed sanpling (AS) were experimented along with complex-task-first, joint and individual task training methods. Gradient surgery (Yu et al. (2020)) was also experimented with joint training to reduce the gradient conflicts between different tasks. Unlike in the literature where multi-task learning are shown to improve the performance of the fine-tuning tasks, in this case, the experiments yielded comparable performance only for the paraphrase detection task. Finally, an ensemble of the different approaches was aggregated and it gave the best performance among all other configurations.

3 Related Work

BERT (Devlin et al. (2019)) uses a stack of bi-directional encoders from transformers (Vaswani et al. (2017)) and is pre-trained with unlabelled data on two unsupervised tasks: Masked Language Model and Next Sentence Prediction. The architecture of BERT is designed to create deep bidirectional representations by conditioning on both left and right context on all layers during pre-training. The BERT model obtained state-of-the-art performance on natural language inference and question-answering tasks by fine-tuning with just one additional task specific layer on top of the BERT embeddings. BERT can handle a pair of sentences by separating them with another special token *[SEP]*, however, this cross-encoder architecture involves passing every possible sentence-pair through all the encoder layers during inference.

SentenceBERT (Reimers and Gurevych (2019)) presents a modified variant of pre-trained BERT network by introducing a siamese network architecture (bi-encoders) to derive meaningful sentence embeddings. For tasks that involve comparing a pair of sentences, the proposed architecture passes each sentence through the siamese network, and concatenates the embeddings for a classification task or passes it through a cosine similarity layer for a similarity detection task. The paper also introduces a pooling layer over the output embeddings of all tokens in the final layer of the BERT network. The bi-encoder architecture proposed in this paper not only reduces the computational complexity of the BERT model but also shows comparable performance on sentence similarity tasks.

More recent work such as (Gao et al. (2021)) has shown that superior sentence embeddings can be obtained by further pre-training a BERT model with a contrastive loss function. This contrastive pre-training method aligns the embeddings of the pre-trained BERT model in such a way that the embeddings of tokens that appear together in the input sentences move close to each other in the embedding space. This paper has also shown empirically that contrastive pre-training improves the performance of BERT model on similarity detection tasks.

While all the previous work in this section discussed architectural changes and training methodology to improve the fine-tuning performance of the pretrained BERT model, Haoming et al. (2019) discusses a regularization method to reduce the overfitting on the downstream tasks and improve generalization on the unseen data. The method proposed in this work, Smoothness Induced Adversarial Regularization causes the pre-trained model to stay robust to aggressive fine-tuning on the training data of the downstream tasks. This efficient fine-tuning helps the pre-trained models attain better generalization.

Several multi-task learning methods were also developed to improve the downstream performance of the pre-trained BERT model. Bi et al. (2022) introduced a multi-task architecture by adding together the individual loss functions of category classification and named entity recognition tasks. Since the size of training data of each individual task may vary, Stickland and Murray (2019) experimented different sampling strategies including proportional sampling, square root sampling and annealed sampling where the latter technique samples more from the larger dataset in the initial epochs. The later epochs are more uniformly trained among each individual task to avoid interference between tasks. Liu et al. (2019) proposed an individual task training strategy where for each iteration, a batch of training data are sampled from a single task and the multi-task network is trained for the sampled task. To further improve the performance of multi-task models, Yu et al. (2020) proposed a method, Gradient Surgery, to reduce the gradient interference among different tasks by projecting the gradient of the i^{th} task onto the normal plane of another task's gradient.

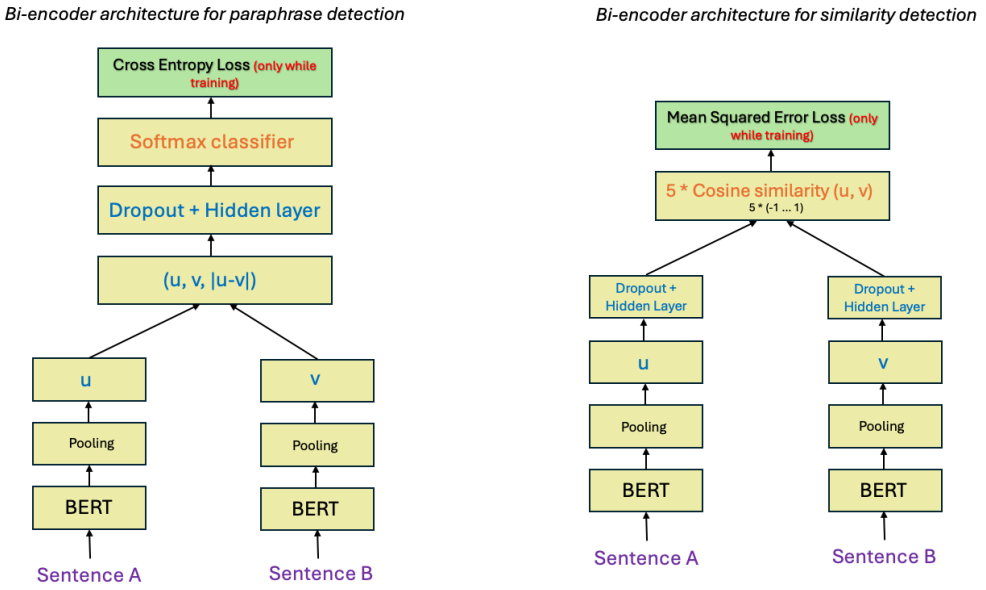

Figure 1: Illustration of single task bi-encoder architecture for paraphrase (left) and similarity (right) detection tasks. *u* and *v* are sentence embeddings from Sentences A and B respectively. For similarity detection task, Cosine similarity of u and v will be used to compute similarity score, which is then scaled by 5 to match the possible range of target values. Cross Entropy Loss is used for other classification tasks.

While all the existing work (except SentenceBERT) in this section uses a cross-encoder architecture, this project uses the bi-encoders as the main building blocks of the neural network.

4 Approach

4.1 Baseline

The baseline is chosen as the pre-trained minBERT model (implemented in part 1 of the project) with task-specific layers on top of the embeddings from its final layer. For sentiment classification task, the task-head contains a softmax layer over the context embedding (embedding of *[CLS]* token). Since the paraphrase detection task has more training datapoints, the task-head contains a binary classifier with one hidden layer. For similarity detection task, a cosine similarity layer over the context embeddings is used as the baseline. For both paraphrase and similarity tasks, each sentence is passed separately through the pre-trained minBERT model (bi-encoders) to obtain the context embeddings. The code for baseline models are developed by the student with the help of starter code provided for this project.

4.2 SentenceBERT (SBERT)

The main architecture is a Siamese network Reimers and Gurevych (2019) where each sentence is passed separately through a pre-trained BERT network and pooling is applied to the dropped out embeddings from the final hidden layer. Figure 1 shows the architecture for the paraphrase and similarity detection tasks. The task-head of the paraphrase detection network consists of a concatenation step that combines the sentence embeddings of the input sentences along with a difference vector between the embeddings. This step is followed by a dropout and a hidden layer with activation before passing the output through a linear layer. For similarity detection, the task-head consists of a pooling layer for the sentence embeddings obtained from the BERT encoder which are then passed through a cosine similarity layer. The task-head for the sentiment classification consists of softmax classifier following a hidden fully connected layer, dropout and activation. The code for the SBERT architecture is developed by the student.

4.3 SimCSE

Further pre-training of BERT is performed with the downstream task's training data using simple contrastive learning (SimCSE) framework (Gao et al. (2021)). Contrastive learning aims to learn effective representation by pulling semantically close neighbors together and pushing apart non-neighbors. In an unsupervised SimCSE architecture, the input sentence is passed through the pre-trained BERT network twice and the contextual embeddings (embedding of *CLS* token) are trained with a cross entropy loss function that uses the other in-batch sentences as the negatives for training. Mathematically, for N sentences

$$l_i = -log \frac{e^{sim(h_i^{z_i}, h_i^{\dot{z_i}})/\tau}}{\Sigma_{j=1}^{N} e^{sim(h_i^{z_i}, h_j^{\dot{z_j}})/\tau}} \tag{1}$$

where $h_i^z = f_\theta(x_i, z)$ is the contextual embedding for sentence i with random mask z for dropout. For single-task fine-tuning, contrastive learning is applied for each task's training data. In the multi-task setting, contrastive learning is applied to the training data for all three downstream tasks. The SimCSE training is implemented by the student.

4.4 Smoothness Inducing Adversarial Regularization

When fine-tuning a large pre-trained language model on a smaller downstream dataset, aggressive fine-tuning causes overfitting of the fine-tuned model to the training data of downstream tasks. To control the model complexity at the fine-tuning stage and and to improve the generalization performance of the fine-tuned model, a smoothness-inducing adversarial regularization method (Haoming et al. (2019)) was applied to the training loss function. For a model $f(. : \theta)$ with n data points of the target task denoted by $(x_i, y_i)_{i=1}^n$, the regularization method solved the following optimization:

$$min_\theta F(\theta) = L(\theta) + \lambda_s R_s(\theta) \tag{2}$$

where $L(\theta)$ is the loss function defined as $L(\theta) = \frac{1}{n}\Sigma_{i=1}^n l(f(x_i; \theta), y_i)$ and $l(.; \theta)$ is the loss function of the target task, $\lambda_s > 0$ is a tuning parameter to weigh the contribution of the adversarial loss $R_s(\theta)$ and $R_s(\theta)$ is the smoothness-inducing adversarial regularizer. R_s is defined as

$$R_s(\theta) = \frac{1}{n}\Sigma_{i=1}^n max_{||x_i^\sim - x_i||_p \leq \epsilon} l_s(f(x_i^\sim; \theta), f(x_i; \theta)) \tag{3}$$

For sentiment classification and paraphrase detection tasks, l_s is choosen as the symmetric KL-divergence, $l_s(P, Q) = D_{KL}(P||Q) + D_{KL}(Q||P)$. For similarity detection task l_s is defined as, $l_s(P, Q) = (P - Q)^2$ where $P = f(x_i^\sim; \theta)$ and $Q = f(x_i; \theta)$. The implementation of SMART loss is adapted from sma which is implemented for a cross-encoder or single sentence architecture. The code is modified to handle the bi-encoder architecture applied in this project.

4.5 Multi-task Learning

Multi-task learning is employed to leverage useful information from related tasks and achieve simultaneous performance improvement on the downstream tasks. Siamese Architecture with Mean Pooling is used as the shared layer among all 3 subtasks and the losses for sentiment classification ($L_{sentiment}$), paraphrase identification ($L_{paraphrase}$) and similarity detection ($L_{similarity}$) are added together to obtain the total loss (Bi et al. (2022)):

$$L_{multitask} = L_{sentiment} + L_{paraphrase} + L_{similarity} \tag{4}$$

4.5.1 Sampling Strategy

Since the training data size is different among all the downstream tasks, different sampling techniques were experimented: 1) minimum-size round-robin (RR) cycle, which randomly samples from each task until the smallest training dataset is exhausted; 2) hybrid-sampling (HS) cycle which samples from the largest training dataset (paraphrase identification) until exhausted for the first n epochs and performs minimum-size round-robin cycle for rest of the training epochs; 3) annealed sampling

(Stickland and Murray (2019)) which samples a batch from a task i with a probability p_i where $p_i \propto N_i^{\alpha}$. α is computed for each epoch e with a total of E epochs as follows:

$$\alpha = 1 - 0.8\frac{e-1}{E-1} \tag{5}$$

4.5.2 Training Schedule

Task scheduling determines the order of tasks on which an MTL model is trained. Different training methods including *joint* training where all three downstream tasks are trained together, was experimented. Other schedules experimented were *individual-task* training (Liu et al. (2019)) combined with annealed sampling, and *max-data-task-first* training where the task with largest training samples (paraphrase detection) are trained for the first n epochs and rest of the epochs are jointly trained with other tasks. The code for all sampling strategies, training schedules and the multi-task architecture were developed by the student.

4.5.3 Gradient Surgery

To reduce the gradient interference among different tasks (which happens when the gradients of different tasks are larger than 90°), Gradient Surgery (GS) as proposed in Yu et al. (2020) is performed to project the gradient of *i*-th task g_i onto the normal plane of another task's gradient g_j that has conflicting gradient: $g_i = g_i - \frac{(g_j \cdot g_i)}{||g_j||^2} . g_j$. The code to perform gradient surgery was adopted from an existing implementation in pytorch (pcg).

4.6 Ensemble

To leverage the benefits of different single-task and multi-task modeling approaches, a model consisting of the best performing models on each task is ensembled. For the sentiment classification and paraphrase detection tasks, the predicted class from the ensemble is based on the absolute majority of the individual models' outputs. For similarity detection task, the ensemble prediction is taken as the average of the predicted similarity of the individual models.

5 Experiments

5.1 Data

For sentiment analysis, SST dataset consists of 8,544 samples for training, 1,101 for development and 2,210 for testing. Quora question pair dataset consists of 283,010 examples for training, 40,429 for development and 80,859 for testing and is used in the paraphrase detection task. SemEval STS benchmark dataset (consists of 6,040, 863, 1,725 examples for train, dev and test sets respectively) is used for similarity detection task.

5.2 Evaluation method

Sentiment classification and paraphrase detection tasks are evaluated on their dev set and test set using accuracy to measure the proportion of samples the models have predicted correctly. Pearson correlation coefficient measures the linear correlation between true similarity and predicted similarity and is used to evaluate the similarity detection task. For the project leader board, the overall score is computed as follows:

$$Overall\_score = \frac{sentiment\_accuracy + paraphrase\_accuracy + \frac{similarity\_pearson\_correlation+1}{2}}{3} \tag{6}$$

5.3 Experimental details

All the model variants are trained on Google Colab Pro with a T4 GPU. Batch size of 8 was used for multi-task training to fit the model and the training data into the GPU memory while for other tasks the training batch size was 16. Adam is the optimization algorithm used to update the model

parameters with a learning rate of 2e-5, and weight decay of 0.01 used in multi-task training. A dropout of $p = 0.3$ is used in the task-head layers. The table below lists the model configurations, training type, training epochs and their training times.

Model Id	Model Type	Architecture	Training Type	Training Epochs	Epoch train time (min)
M1	Baseline (Sentiment)	BERT + Softmax Layer	Last layer	10	1.05
M2	Baseline (Paraphrase)	BERT + 2-layer classifier	Full model	10	50.85
M3	Baseline (Similarity)	BERT + Cosine similarity	Full model	10	2.94
M4	BERT + MeanPool (Sentiment)	BERT + Pool + 1-Hidden Layer	Full model	10	1.23
M5	Bi-Encoder (Paraphrase)	BERT + Pool + Concat + 1-Hidden Layer	Full model	5	56
M6	Bi-Encoder (Similarity)	BERT + Pool + 1-Hidden Layer + Cosine similarity	Full model	10	3.2
M7	MTL: Bi-Encoder - RR - JT	BERT + Pool + Task Specific Layers	RR + JT	10	35.1
M8	MTL: Bi-Encoder + HS	BERT + Pool + Task Specific Layers	HS + Max-data-first	7 (3 max data)	71.6
M9	MTL: Bi-Encoder + MTSimCSE + HS	BERT + MTSimCSE + Pool + Task Specific	Hybrid-sampling	10 (4 max data)	70.6
M10	MTL: Bi-Encoder + MTSimCSE + AS	BERT + Pool + Task Sepcific	AS + Individual Task	3	56.3
M11	BERT + MeanPool + SimCSE (Sentiment)	BERT + Pool + 1-Hidden Layer	Full model	10	2.2
M12	Bi-Encoder + SimCSE (Paraphrase)	BERT + Pool + Concat + 1-Hidden Layer	Full model	6	60.6
M13	Bi-Encoder + SimCSE (Similarity)	BERT + Pool + 1-Hidden Layer + Cosine similarity	Full model	25	5.02
M14	Bi-Encoder + SimCSE + SMART (Sentiment)	BERT + Pool + 1-Hidden Layer	Full model	5	2.07
M15	Bi-Encoder + SimCSE + SMART (Paraphrase)	BERT + Pool + Concat + 1-Hidden Layer	Full model	6	58.1
M16	Bi-Encoder + SimCSE + SMART (wt=1) (Similarity)	BERT + Pool + 1-Hidden Layer + Cosine similarity	Full model	10	4.1
M17	Bi-Encoder + SimCSE + SMART (wt=3) (Similarity)	BERT + Pool + 1-Hidden Layer + Cosine similarity	Full model	10	4.1
M18	Bi-Encoder + MTSimCSE + SMART (Sentiment)	BERT + Pool + 1-Hidden Layer	Full model	5	2.07

5.4 Results

Table 1 lists the results of each model variant on their development set. The bi-encoder architecture clearly improves the baseline performance, on all tasks for both single-task and multi-task settings. Additional pre-training based on contrastive learning framework improves the model performance for single-task models and with adversarial regularization, the generalization improves. The adversarial loss adds more value to the tasks that had smaller training samples, improving the accuracy by 1.9 percentage points for sentiment classification task, and correlation by 1.3 points for the similarity detection task. For multi-task models, training max-data-first strategy leads to better overall performance than joint and individual training schedules. Based on the experimental results, the bi-encoder network is a good architecture to improve the performance of BERT on downstream tasks.

Model Id	SST Accuracy	QQP Accuracy	STS Correlation	Overall Score
M1	41.4	-	-	-
M2	-	66.3	-	-
M3	-	-	63.5	-
M1, M2, M3	-	-	-	62.7
M4	51.3	-	-	-
M5	-	*88.5*	-	-
M6	-	-	75.4	-
M4, M5, M6	-	-	-	75.8
M7	50.7	81.6	71.7	72.7
M8	47.2	*88.1*	76.1	74.5
M9	46.4	88.1	77.2	74.4
M10	49.0	84.8	71.0	73.1
M11	*51.7*	-	-	-
M12	-	*88.8*	-	-
M13	-	-	*81.0*	-
M11, M12, M13	-	-	-	77.0
M14	*53.3*	-	-	-
M15	-	**88.9**	-	-
M17	-	-	**82.3**	-
M14, M15, M17	-	-	-	77.8
M16	-	-	*81.5*	-
M18	**53.6**	-	-	-
Ensemble	**54.2**	**90.2**	**84.8**	**79.0**

Table 1: Performance on development set for the model variants. Adding contrastive pre-training and adversarial regularization improves the performance on all 3 tasks. Single-task models outperform multi-task models on all the downstream tasks. Ensemble model performs better than other variants in the development set

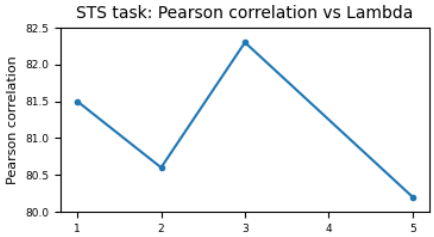

Figure 2: Dev set performance of STS task for different Adversarial loss weights λ

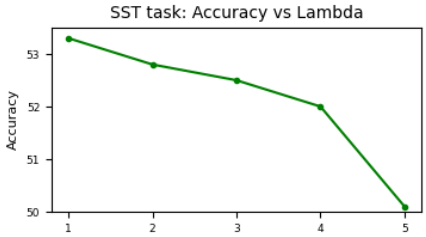

Figure 3: Dev set performance of SST task for different Adversarial loss weights λ

The ensemble model for each task consists of the following models: Sentiment - M11, M14, M18; Paraphrase - M5, M8, M12, M15; Similarity - M13, M16, M17. The ensemble pushes the overall score by 1.2 points for the development set and it is the final model submitted to the test set leaderboard. The test set scores for each task and the overall score are shown in Table 2.

Task	Score
Sentiment	55.0
Paraphrase	90.1
Similarity	81.5
Overall score	78.6

Table 2: Test set performance of the ensemble model. The final prediction in the sentiment and paraphrase tasks are based on the majority of predicted classes of the individual models. For similarity detection, the ensemble prediction is the average of predictions of the individual models.

6 Analysis

Single-Task vs Multi-Task Learning: Single-task models performed better than multi-task models in all three tasks. Especially for SST and STS tasks, performance of the multi-task models dropped significantly. One reason for this is the imbalance in the training dataset among the tasks. The training data for QQP is over 45 times more than that of the smallest training dataset (STS) and using the smallest dataset size underfits the QQP task while utilizing all QQP data overfits the other two tasks.

SimCSE and Adversarial Regularization: Pretraining the bi-encoders with SimCSE acts as a form of regularization as the contrastive learning objective eases the anisotropy problem in the pre-trained embeddings by pushing the negative instances apart. Adversarial regularization also reduces the aggressive updation of the model during the fine-tuning stage by injecting a perturbation to each training cycle. Figures 2 and 3 show the dev set performance for different weights (λ) of the adversarial loss. The behavior of the model performance is not monotonic with λ for STS task and based on grid-search, $\lambda = 3$ yielded the best result.

Pooling and Activation Ablation: As recommended in Reimers and Gurevych (2019), MEAN pooling was used for all the model variants. However, *CLS* pooling was also evaluated for all three tasks in a SimCSE pre-trained single-task setting and their performances are shown in Table 3. From the results, it is observed that *MEAN* pooling clearly improves the performance of the bi-encoder model for STS and QQP tasks while the type of pooling do not have large effect on SST task. For STS task, the ReLU activation at the last layer was replaced with sigmoid (also removes negative values) and from the results in Table 4, using sigmoid causes huge performance drop on the STS task.

Error Analysis: Like any other model, the models discussed in this work is not perfect and has its limitations. Figure 4 shows the confusion matrix for SST task. From the matrix, it is observed that the model does not distinguish very well between negative and somewhat negative sentences, and positive and somewhat positive sentences. For example, the model is more likely to predict a sentence as negative if it only sees words that are used in the negative context such as the example *unflinchingly bleak and desperate* (actual: *somewhat negative*, model: *negative*). Also, the model does not handle the negation well and tends to predict a class higher when a negation prepends a positive word such

Table 3: Dev set performance for CLS pooling

Task	Performance
Sentiment (SST)	51.9
Paraphrase (QQP)	77.2
Similarity (STS)	70.9

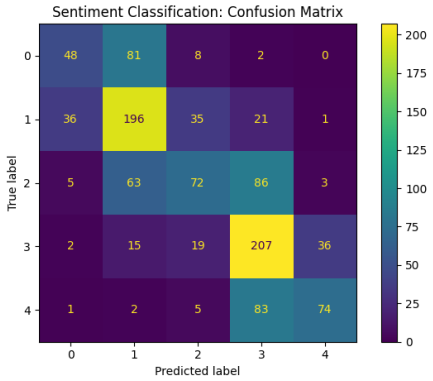

Figure 4: SST Confusion matrix

Table 4: Dev set performance for STS task with sigmoid activation

Model	Correlation
SimCSE	59.2
SimCSE + Adversarial Reg.	72.4

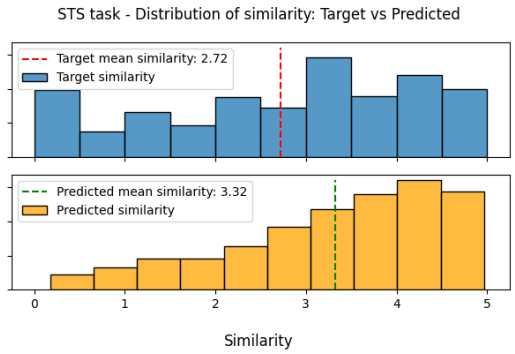

Figure 5: STS distribution. Predicted similarity are skewed towards higher similarity values. Average predicted similarity is higher than the average target similarity

as *the humor is n't as sharp , the effects not as innovative , nor the story as imaginative as in the original* (actual: *negative*, model: *somewhat negative*). For STS task, the main difference in the model behavior is the that the predicted similarites are a little skewed towards higher similarity values, as shown in Figure 5. One reason for this could be the constant (value of 5) multiplied to the output of the final layer of the STS neural network.

7 Conclusion

The bi-encoder architecture applied in this project improves the performance of the vanilla BERT model on sentiment, paraphrase and similarity detection downstream tasks. It is also learnt from the experiments that pre-training the BERT model via contrastive learning and adding adversarial regularization improves the overall model performance. An ensemble of the methods gave the best performance and yielded a score of 78.6% on the test set. The primary limitation of the sentiment detection models are their ability to differentiate sentences that contains multiple negations, and the skewness of the similarity detection model towards a higher similarity. The immediate scope for future work would be to apply a different optimization technique such as Bergman proximal point optimization to optimize adversarial loss, to compare the unsupervised SimCSE applied in this project with a negative ranking loss function in a supervised setting (using non-sample sentences as negatives) and to use a triplet network with out-of-sample negatives and triplet loss for similarity detection models.

8 Ethics Statement

The project uses pre-trained weights of the BERT model to fine tune on the downstream tasks. Hence, any bias in the BERT model will get carried over to the fine-tuned model which might pose a risk of giving different outputs for the same sentence based on a protected characteristic such as gender (say if the model exhibits gender bias and is deployed for a sentiment analysis, then the model's predictions might differ based on the genders used in the sentences). Another societal issue is that the model does not refrain from giving an output for sensitive input sentences (eg. sentences containing sensitive comments towards protected classes like race, gender, ethnicity etc.). To overcome the latter issue, sentences containing sensitive tokens can be blocked before feeding them into the model. Bias

in the pre-trained models can be reduced by training on diverse set of examples during fine-tuning. Examples that contain sensitive information or any forms of bias can be removed from the fine-tuning training data to avoid further increasing the bias in the final model. Also, the model needs to be tested rigorously for bias and fairness before being deployed for real-world consumption.

References

https://github.com/archinetai/smart-pytorch/tree/main.

https://github.com/weichengtseng/pytorch-pcgrad/blob/master/pcgrad.py.

Qiwei Bi, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Hanfang Yang. 2022. MTRec: Multi-task learning over BERT for news recommendation. In *Findings of the Association for Computational Linguistics: ACL 2022*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*.

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

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*.

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*.