SMART loss vs DeBERTa

Stanford CS224N Default Project

Michael Hayashi
Department of Computer Science
Stanford University
mikehaya@stanford.edu

Michael Liu
Management Science and Engineering
Stanford University
zhenliu@stanford.edu

Roberto Lobato
Management Science and Engineering
Stanford University
rlobato@stanford.edu

Abstract

Our project proposal is designed to maximize the multitask prediction accuracy of natural language processing (NLP) models through the implementation and extension of the BERT architecture and its iteration: DeBERTa. Initially targeting to outdo the performance benchmarks set by the standard minBERT model in critical NLP tasks such as sentiment analysis, paraphrase detection, and textual similarity, our journey began with less than promising results. By incorporating strategies such as SMART loss application, employing early-stopping techniques, altering the processing sequence, and meticulous fine-tuning, exponential learning decay, and dropout we managed to substantially elevate our model's capabilities, achieving a overall test accuracy of 0.778. Contrary to our initial hypotheses, refinements on top of BERT were better compared to direct changes to the original architecture like in DeBERTa.

1 Disclaimers

CS 224N Mentor: Andrew Hyungmin Lee. We do not have external collaborators or mentors. The project is entirely part of CS 224N.

2 Introduction

The field of Natural Language Processing (NLP) has been categorized for its fast-paced evolution in the last couple of years. Still, the development and refinement of models capable of understanding, interpreting, and generating human language remains an important challenge. One of the core advancements is the Transformers architecture in deep learning and its implementation, for example in the BERT (Bidirectional Encoder Representations from Transformers) model. These models have revolutionized our approach to a wide array of NLP tasks, including sentiment analysis, paraphrase detection, and semantic textual similarity assessment. However, despite their remarkable successes, these models still face significant hurdles in terms of generalization, efficiency, and understanding of nuanced human language complexities.

In our present work, we address these challenges by implementing, examining, and fine-tuning the architecture of BERT and its update, DeBERTa (Decoding-enhanced BERT with disentangled attention). BERT has set a new standard in NLP by pretraining deep bidirectional representations from unlabeled text, significantly improving performance across various NLP tasks. DeBERTa further enhances BERT's architecture by introducing a disentangled attention mechanism, which separately models the content and position of words, and by integrating an enhanced mask decoder for more

effective pretraining. These advancements aim to refine the model's understanding of language nuances, improve generalization to unseen data, and enhance interpretability. Our modifications have led to significant improvements in model performance, as evidenced by reaching an overall test accuracy of 0.778.

Moreover, our experiments highlight the critical importance of model tuning and the choice of loss functions in achieving optimal performance. By adjusting these aspects, we were able to overcome initial setbacks, such as models defaulting to predict a single class for all samples or suffering from overfitting.

3 Related Work

In the realm NLP, models like BERT have marked a significant milestone, offering great capabilities in understanding, interpreting, and generating human language. Despite their success, these models have encountered limitations, particularly in differentiating the importance of a word's content from its position within a sentence, a challenge that directly impacts their ability to grasp the nuanced semantics of language. (Devlin et al., 2018)

Addressing this issue, He et al. (2021) introduced DeBERTa (Decoding-enhanced BERT with disentangled attention), an iteration of BERT that innovates with a disentangled attention mechanism. This mechanism represents each token with separate vectors for content and relative position, allowing for a more granular computation of attention scores across four dimensions: content-to-content, content-to-position, position-to-content, and position-to-position. Such an approach promises a more nuanced understanding of word interrelations, enhancing the model's capacity to process complex linguistic structures. (He et al., 2023)

Further enriching this landscape, Jiang et al. (2020) contributed techniques aimed at improving the generalization and stability of these models. Smoothness-Inducing Adversarial Regularization (SIAR) introduces an adversarial component to the loss function, encouraging the model to be less sensitive to minor variations in input and, therefore, more generalizable to new data. Meanwhile, Bregman Proximal Point Optimization (BPPO) offers a method to temper the training updates, ensuring a smoother, more stable convergence by establishing a "trust region" that limits how far each update can deviate from the current parameter set. This technique is especially crucial in preventing overfitting, a common pitfall when fine-tuning large pre-trained models on smaller, task-specific datasets. (Foret et al., 2020)

4 Approach

In the original BERT architecture, each word is represented by a single vector which contains information about the word content and position in the form of the element-wise embedding sum. The disadvantage of this approach is potential information loss: the model might not differentiate whether a word itself or its position gives more importance to a certain embedded vector component. For example, the words "deep" and "learning" are much more dependent when they appear next to each other than in different text parts. This justifies why it is necessary to consider content-to-position relations. To fix that, He et al. (2021) introduced DeBERTa with two novel techniques. The first is the disentangled attention mechanism, where each token at position i is represented with two vectors, $\{H_i\}$ and $\{P_{i|j}\}$ which represent its content and relative position with the token at position j. The calculation of the cross-attention score between tokens i and j can be decomposed into four components as

$$A_{i,j} = \{H_i, P_{i|j}\} \times \{H_j, P_{j|i}\}^T = H_i H_j^T + H_i P_{j|i}^T + H_j P_{i|j}^T + P_{i|j} P_{j|i}^T \tag{1}$$

That is, the attention weight of a word pair can be computed as a sum of four attention scores using disentangled matrices on their contents and positions as content-to-content, content-to-position, position-to-content, and position-to-position.

The second innovation is an enhanced mask decoder which uses absolute positions in the decoding layer to predict the masked tokens during pre-training. The local context alone is insufficient for the model to distinguish between the masked words **store** and **mall** in the sentence "a new **store**

opened beside the new **mall**" because both words follow "new" with the same relative positions. To address it, the model must consider absolute positions, as the sentence subject is "store," not "mall." Syntactical nuances like these heavily rely on the words' absolute positions within the sentence.

Architecture is not all you need to predict in downstream tasks. The training procedure can also be very important. To improve the generalization capabilities, we utilized 2 techniques from Jiang et al. (2020). The first technique is Smoothness-Inducing Adversarial Regularization. The idea is to keep the parameters in a space where small perturbations do not lead to significant changes in the output of the model under the assumption that if it is less sensitive to minor input variations, then it is more likely to generalize well to new data (Figure 1). To achieve it, an adversarial regularization term is added to the model's loss function. By minimizing this term along with the original loss function, the model is encouraged to produce similar outputs for both perturbed and unperturbed inputs, hence enforcing smoothness and avoiding sharp minima.

$$\text{SIAR}(\theta) = L(\theta) + \lambda_s R_s(\theta)$$

The SIAR loss is the original loss plus a regularization hyperparameter times the adversarial regularize R_s. Let δ_i be the perturbation added to input x_i, then

$$R_s(\theta) = \frac{1}{n} \sum_{i=1}^{n} \max_{\|\delta_i\|_p \leq \epsilon} d_{KL}(f(x_i + \delta_i; \theta), f(x_i; \theta))$$

where d_{KL} is the KL-divergence.

The second technique its called Bregman Proximal Point Optimization. It is an optimization method designed to prevent aggressive updates during the model training process, thereby stabilizing the fine-tuning of pre-trained models for specific tasks. The main idea behind Bregman is to create a "trust region" around the current parameters so that the new update does not stray too far from the previous set of parameters. This approach tries to ensure a smoother progression towards the optimum. The technique utilizes the Bregman divergence, a generalized measure of distance between two points in the parameter space defined in terms of a strictly convex functions. At each step of the optimization, the method solves a subproblem that minimizes the original objective function regularized by a term based on the Bregman divergence. Pre-trained models are typically large and trained on vast amounts of data. When fine-tuning such models on a smaller, task-specific dataset for downstream tasks, there's a risk of overfitting. By stabilizing the fine-tuning process, this optimization method, we hope to avoid overfitting or instability, and preserve the generalization ability of the model.

$$\theta^{(t+1)} = \arg\min_{\theta} \left\{ F(\theta) + \mu D_\phi(\theta, \theta^{(t)}) \right\}$$

Where $D_\phi(x, y)$ is the Bregman distance associated. It is the difference between the evaluations at points x and y and the value of the first-order Taylor expansion of ϕ around the points:

$$D_\phi(x, y) = \phi(x) - \phi(y) - \langle \nabla\phi(y), x - y \rangle$$

with as $\langle \cdot, \cdot \rangle$ the dot product.

To address the low accuracy in our models' ability to assess sentiment analysis, we recurred to data augmentation. By introducing additional training datasets, we were hoping to broaden our model's exposure to more data patterns and structures. We were able to do this by including the Amazon Movies and TV Review dataset.

5 Experiments

5.1 Data

We used the three default datasets to train and evaluate the models, plus some experiments with data augmentation using a fourth dataset.

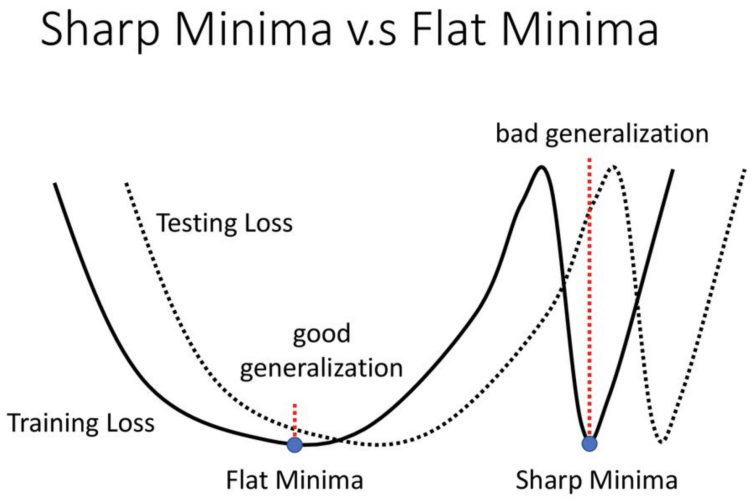

Figure 1: Generalizing with Sharp minima vs Flat Minima.

1. Stanford Sentiment Treebank (SST): It provides sentences from movie reviews and their sentiment labels (positive or negative) and will be used for sentiment analysis.
2. Quora Question Pairs (QQP): Contains pairs of questions with labels indicating whether the questions are paraphrases of each other and is used for duplicate detection.
3. Semantic Textual Similarity Benchmark (STS): Used for semantic textual similarity tasks, it comprises pairs of sentences with similarity scores, indicating the degree of semantic similarity between the sentences.
4. Amazon Movies and TV Review dataset [1]: 1,697,533 reviews from 1 to 5 stars of Movies and TV shows on Amazon spanning May 1996 - July 2014. We preprocessed the data to have an equal split of 1 to 5 star reviews and review length of <200 characters for sentiment prediction training.

5.2 Evaluation method

We evaluate the default BERT model and our improved versions across the three datasets. For SST, we use accuracy to assess the model's ability in sentiment analysis. With the Quora dataset, we also evaluate the accuracy, reflecting its capacity to identify paraphrases accurately. For STS, the evaluation focuses on the Pearson correlation, measuring the model's performance in capturing semantic similarity between sentences.

5.3 Experimental details

Our first version of BERT wasn't performing well initially, with a Sentiment accuracy (SST) of 0.478, a paraphrase accuracy (Quora) of just 0.378 and a Semantic Textual Similarity (STS) correlation of 0.558. We consider this to be our baseline for comparisons (BERT v1 in Table 1). Recognizing the need for rapid progress, we couldn't afford to waste too much time for the model to finish training on the entire dataset before beginning the fine-tuning process and exploring other strategies. Therefore, we opted to train on 1/20th of the paraphrase dataset to match the dataset size of the Stanford Sentiment Treebank (SST) and STS datasets.

To significantly increase our model's accuracy, we implemented two important adjustments. Initially, our approach involved running the forward pass on both inputs separately and then concatenating the results. Changing our strategy to concatenate the inputs first and then perform the forward

[1] https://jmcauley.ucsd.edu/data/amazon/

pass proved to be very useful (BERT + new predict in Table 1). Additionally, we noticed that our paraphrase dataset often resulted in a consistent loss of 0.378, indicating that the model was inaccurately predicting a constant value of 1 for all examples. To address this, we switched from using the binary cross-entropy loss function to the nn.BCEWithLogits loss function with reduction='sum' . This adjustment significantly enhanced our model's accuracy, achieving 0.713 on the development set and 0.98 during training (BERT + new predict + BCE Logits Loss in Table 1).

The large gap between the development accuracy and training accuracy hinted at a possible issue with overfitting. To counter this, we introduced regularization through the implementation of Smoothness-Inducing Adversarial Regularization (SIAR). The idea is that constraining the search space to a flat minima, will let us generalize better while reducing the chance of overfitting. Indeed, this single change marked our biggest jump in improvement, elevating our accuracy to 0.764 (BERT + SIAR in Table 1).

In parallel, we developed our implementation of the DeBERTa architecture. Initially, our DeBERTa model improved the performance on the Quora and STS benchmarks but didn't enhance the results for SST compared to our baseline, achieving an overall accuracy of 0.651. Following this, we applied additional fine-tuning and integrated SIAR to constrain the search space. Surprisingly, this approach resulted in a lower overall accuracy of 0.71, compared to our BERT model with SIAR, which reached an accuracy of 0.76. This led us to concentrate our efforts on further improving the BERT model.

We fine-tuned the model using the full paraphrase dataset. After training for 3 hours across 10 epochs, the model achieved an accuracy of 0.764. Noticing that the accuracy peaked at the third epoch before starting degrading, hinted at a too high learning rate. We experimented with different values of the learning rate. By reducing the learning rate to 1e-6, we had a minor change in accuracy to 0.765. We concluded this experimentation phase, by opting to stop the process early and settling on a learning rate of 1e-5. This approach resulted in a development accuracy of 0.77.

Continuing the implementation of the SMART Loss, we added the second technique called Bregman Proximal Point Optimization (BPPO) to reduce the aggressive updates we were having during the model training process. This implementation (BERT + BPPO) led to modest increments, but using the similar early stop procedure as before led to a higher total accuracy (0.771). We realized that our model often reached peak dev accuracy after a handful of epochs before falling again, and switched over to an exponentially decaying learning rate to prevent overfitting and overshooting of the minima.

We observed that none of our strategies significantly improved the sentiment accuracy. To address this, we added an extra training dataset from Amazon Movies and TV Reviews. Unfortunately, the accuracy actually decreased for our target metric, even after we added the BPPO implementation.

Finally, we attempted different dropout values to further decrease overfitting in our model. We found that dropout values of 0.4 and 0.5 improved over a dropout probability of 0.3, with 0.4 + early stopping having the best performance (0.4 dropout in Table 1). At last, our highest total development accuracy of 0.776 was achieved by using BERT with BPPO, Exponential Decay Learning rate, and 0.4 dropout with 9 epochs.

5.4 Results

From baseline BERT we achieve the following results:

- SST (Stanford Sentiment Treebank) dev accuracy: 0.478
- Paraphrase dev accuracy: 0.378
- STS dev correlation: 0.558
- Overall dev score: 0.478

Our best fine-tuned model resulted in:

BERT + BPPO + ExpDecayLr + 0.4 dropout + 9 epochs

- SST (Stanford Sentiment Treebank) test accuracy: 0.506
- Paraphrase test accuracy: 0.892
- STS test correlation: 0.869
- Overall test score: 0.778

Model	SST	Quora	STS	Total Acc
BERT v1	0.478	0.378	0.558	0.478
BERT + new predict	0.482	0.378	0.856	0.572
BERT + new predict + BCE Logits Loss	0.482	0.800	0.856	0.713
DeBERTa v1	0.414	0.750	0.787	0.651
BERT + SIAR	0.487	0.881	0.848	0.764
BERT + SIAR + lr 1e-6	0.500	0.884	0.834	0.765
BERT + SIAR + 3 epochs	0.499	0.883	0.858	0.77
DeBERTa v2 + SIAR	0.432	0.857	0.825	0.71
BERT + SIAR + 100k Amazon data	0.492	0.887	0.862	0.77
BERT + BPPO	0.473	0.89	0.86	0.764
BERT + BPPO + 4 epochs	0.492	0.885	0.869	0.771
BERT + BPPO + 100k Amazon data	0.468	0.887	0.849	0.76
BERT + BPPO + ExpDecayLr	0.49	0.89	0.867	0.771
BERT + BPPO + ExpDecayLr + 0.4 dropout	0.49	0.89	0.868	0.771
BERT + BPPO + ExpDecayLr + 0.4 dropout + 9 epochs	0.501	0.89	0.871	0.776
BERT + BPPO + ExpDecayLr + 0.5 dropout	0.506	0.887	0.853	0.773

Table 1: Results comparison.

6 Analysis

We attempted improving our model from a variety of angles from changing the model architecture, adding additional training datasets, and improving model generalization. Out of all the methods, improving model generalization with regularization techniques such as SIAR, BPPO, and dropout had the most noticeable improvement on our prediction scores. This makes sense as our model is tasked with learning a multitask approach to simultaneously perform well on three different datasets, and overfitting to one particular dataset comes at the expense of another dataset's performance. Improving model generalization also has the easiest implementation since we can make minor adjustments to the code and directly see their effect on performance.

On the other hand, changing the model architecture from BERT to DeBERTa requires significant updating to various parts of the code base. Failing to properly adjust one area can sabotage the effects of the remaining updates. We expect that our DeBERTa architecture performed slightly worse because we used pre-trained weights from a BERT-based uncase, but we should had used weights from a pre-trained DeBERTa model. The newly added parameters won't be updated until the fine-tuning stage. This issue becomes more apparent when we apply the enhanced make decoder where the position embedding isn't added on the input layer and moved to the output layer.

Another key roadblock to achieving better overall accuracy was the SST dataset. SST performance consistently stayed around or below 0.5 accuracy while both the paraphrase and STS datasets achieved near 0.9 accuracy. In general, the SST is a very hard-to-predict dataset. Predicting sentiments on a 1-5 scale from subjective movie reviews is difficult even for humans to do, compared with predicting the similarity of sentences or paraphrasing: movies are subjective, people might grade differently, and sometimes context is very unique. Also, the SST dataset is trained first, therefore the weights may get rewritten in favor of later datasets. Adopting a new sampling method or combined loss function may have led to a more evenly and non-biased distribution. Implementing an ensemble model may also have improved performance per dataset, albeit at the expense of computational efficiency.

7 Conclusion

Through a series of experiments, including the application of SMART loss, early-stopping, exponential learning decay, dropout techniques, our team was able to address significant challenges to generalizing in downstream tasks such as overfitting. The adoption of these strategies led to a notable improvement in the model's performance, achieving a development accuracy of 0.776 and test accuracy of 0.778, which marks a significant step forward in the endeavor to enhance the capabilities of NLP models.

Our success was not without its challenges. The iterative process of fine-tuning and adjusting the model parameters underscored the complex nature of machine learning models and the delicate balance required to achieve significant improvements in specific tasks. Two of our strongest initial hypothesis turned to be incorrect. The DeBERTa architecture, while helpful in its first iteration, proved difficult to integrate the other changed. The integration of additional training datasets did not help as much as we expected. These results demonstrate the critical importance of model tuning, the choice of loss functions and the need to rapidly iterate over different attempts. Despite the advancements made, we were particularly baffled with the lack of improvement in the sentiment analysis accuracy.

For future explorations, we're thinking about using a round robin sampling method to boost our model's overall performance (Stickland and Murray, 2019). Augmenting the data seemed like the right move, though its execution might not have been the best. The round robin approach would cycle through the data, ensuring all classes are fairly and consistently represented, which might further improve our metrics. We're also considering the possibility of dedicating more time to fine-tuning our DeBERTa model so that it could allow us to more effectively integrate the methods we've applied to BERT.

References

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

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2020. Sharpness-aware minimization for efficiently improving generalization.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization.

Asa Cooper Stickland and Iain Murray. 2019. Bert and pals: Projected attention layers for efficient adaptation in multi-task learning.

A Appendix