Three Heads Are Better Than One

Stanford CS224N Default Project

Esteban Cambronero Saba
Department of Computer Science
Stanford University
ecambr23@stanford.edu

Jesus Meza Rosales
Department of Computer Science
Stanford University
jemeza@stanford.edu

Abstract

Models with multitask capabilities are a key area of focus in the NLP community, this has been expanded by encoders such as BERT. In this paper we explored the impacts of having task specific heads that function on the output of BERT embeddings. We also used gradient surgery and data augmentation in an effort to alleviate the issue of gradient interference. Which can often come as a result of conflicting gradient vectors or a dataset being larger. We tested our implementation on downstream tasks for BERT, specifically: semantic text similarity, paraphrasing, and sentiment analysis.

1 Key Information to include

- Mentor: None
- External Collaborators: None
- Sharing project: No

2 Introduction

Multitask models in the field of natural language processing have recently emerged due to the success of pre-trained models such as BERT, which allow word embeddings to be developed without labeled data. This has drastically changed the landscape due to their effectiveness of capturing the semantic relationships between words. Nevertheless, these embeddings are not sufficient to perform downstream tasks. Instead, the embeddings can be used as the input into other task specifc model heads. We began with a simple approach, for the dual sentence tasks concatenated the bert embeddings and fed it into a linear layer. For the sentiment task we only used a linear layer to process the BERT embeddings. We realized that these model heads were too general and could not capture the relationships between the sentences or the labels. Thus, we opted to iterate on the heads and arrived at the final model that is described below.

Despite having task specific heads, training one model to complete different tasks well, poses great challenges, especially when the datasets for each task are of different sizes. Data augmentation as in Easy Data Augmentation (EDA) [4] offers a solution that when carefully combined with gradient surgery has the potential to produce interesting results.

We used EDA to augment areas of data that we felt could be expanded upon and then used gradient surgery to alleviate the effects of gradient interference. We used accuracy on three datasets (STS, Quora, SST) to generate benchmarks results.

3 Related Work

3.1 Task Specific Heads

There has been substantial work on the utilization of multiple heads with a shared base model both in NLP[2][1] and computer vision [3]. These papers provided baseline ideas that we built upon to develop our models. Additionally, [6] also gave important insight on approaching the semantic text similarity and paraphrase tasks. Specifically, using concatenated sentences before passing them in through Bert to generate a single embedding vector for both sentences. Convolutional neural networks have showed effectiveness in sentiment classification [10]. For this reason we chose to approach the Sentiment classification head with a CNN layer.

3.2 Gradient Surgery

Training a model to perform well on a multitude of tasks is more difficult than training for individual tasks as shown in [9]. One of the reasons is gradient interference, which arises when gradients for different tasks of the model have a cosine similarity less than 0. This prevents the model from properly finding a minimum in the combination of both spaces. Gradient surgery, as described in [5], is an answer to the issue of gradient interference. Gradient surgery functions by projecting the gradient of one task to the null space of the gradient of each other tasks. Then subtracting this projection from the original gradient as follows:

$$g_i^{PC} = g_i^{PC} - \frac{g_i^{PC} \cdot g_j}{\|g_j^2\|} g_j$$

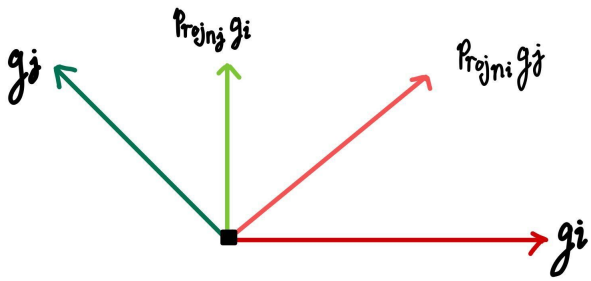

Figure 1: Gradient Projection Example

As seen in 1 this causes the gradients to converge to a point in between when they are conflicting. This method increases the probability that the sum of the gradients can point into the direction of a minimum for all three of the different tasks. In practice this method works by computing the loss and gradients of the model for each specific task. Then iterating through each gradient and applying the formula above as necessary, before summing them together.

3.3 Data Augmentation

Data augmentation is a method that is used to expand datasets that are lacking in quantity but not quality. We based our work on EDA [4] that proposed universal methods of data augmentation for NLP that randomly insert, replace, delete, and swap words within sentences. The proposed changes allowed us to augment the datasets with less data and should help to prevent the model for overfitting to a particular task while maintaining their expressive power.

4 Approach

MinBert was the foundational piece of our model and we added one head for each of the different tasks: sentiment analysis, semantic text similarities, and paraphrase detection. For the sentiment

analysis head, we took the BERT embeddings and fed them into a one dimensional convolution layer with a kernel size of 3. We found that just the convolution outperformed the addition of any linear layers as was done with the other heads. This outputs a batch size by num labels tensor that is our output. For the paraphrase head, we take both sentences, concatenate them, and then pass that into BERT. We then take this combined embedding and pass it into a linear layer, dropout, relu, and another linear layer. We then apply the sigmoid activation function which gives us our output.

For the semantic similarity we do the same procedure except we scale the output vector by five to ensure that the output is within the correct range. We began our baseline with concatenating the two embeddings from BERT in both paraphrase and sentiment. We noticed that this resulted in very low accuracy for both, but particularly for STS. We then scaled the output by five and noticed an improvement so that PARA and STS were equivalent, but Sentiment was further ahead. We then implemented cosine similarity for the sentiment head and this drastically improved performance. Following this, we attempted the same for the paraphrase head but the accuracy fell significantly. We then attempted to improve PARA and capture interactions by using the dot product of the BERT embeddings. This resulted in better learning, but still significantly lower accuracy. Our next attempt to capture interactions was to concatenate the two sentences and run them through a sequence of layers. This improved our accuracy past that of cosine similarity. However, when implementing this on both PARA and STS we ran into conflicting gradients that led to decreasing results for both. Finally, we arrived at the model shown below 2 which boosted our accuracy significantly. We found that varying either PARA or STS head from the other caused an imbalance in learning.

For data augmentation, we followed the guidelines outlined in EDA. Essentially, we would add, swap, delete, and insert words randomly. We processed each sentence in the dual sentence tasks separately and did not change the labels associated with the sentences. This approach allowed us to augment the data substantially and generated nine entries for each one in the original dataset. We did this for the STS and SST datasets to make their sizes more comparable to that of Quora. By doing so we were able to train with the same batch sizes but larger pool of examples per epoch. One important caveat is that this method does have the downside of generating potentially incorrect data. For example, "She was not happy" becoming "She was happy". The sentiment of this sentence has drastically changed, but the label did not. Nevertheless, in our experiments this did not seem to affect the robustness of the model and it is unclear if examples like these were generated.

Initially, we utilized a round robin approach to training for the different tasks. After computing the loss for each of the different tasks, we computed the backward gradient and then took a step in the direction of the gradient. The loss functions that were utilized for each tasks were: mean squared loss for sentiment classification, cross entropy loss for semantic text similarity, and binary cross entropy for our paraphrase tasks. The gradient steps occurred in the order of sentiment classification, paraphrase detection, and semantic text similarity classification.

After achieving baseline we results, we utilized a slightly modified version of PCGrad [7], an existing implementation of gradient surgery based on [5]. We hoped that gradient surgery would reduce the negative effects of gradient interference. After calculating the losses for each tasks they were combined using the PCGrad optimizer before taking a step in the direction of the newly calculated gradient.

5 Experiments

5.1 Data

The datasets that were used were: Quora dataset for paraphrase detection, Stanford Sentiment Treebank dataset for sentiment classification, and the STS benchmark dataset for STS classification. The paraphrase dataset provided quora questions and whether or not one was the paraphrase of another. The SST dataset contained movie reviews and an associated score for sentiment, 1 being negative and 5 being positive. Finally, the STS dataset contained pairs of sentences with a score indicating how similar they were, 1, not similar at all, and 5, they contained the same meaning. These datasets were provided by the course staff and the STS and SST datasets were augmented as described in the approach section.

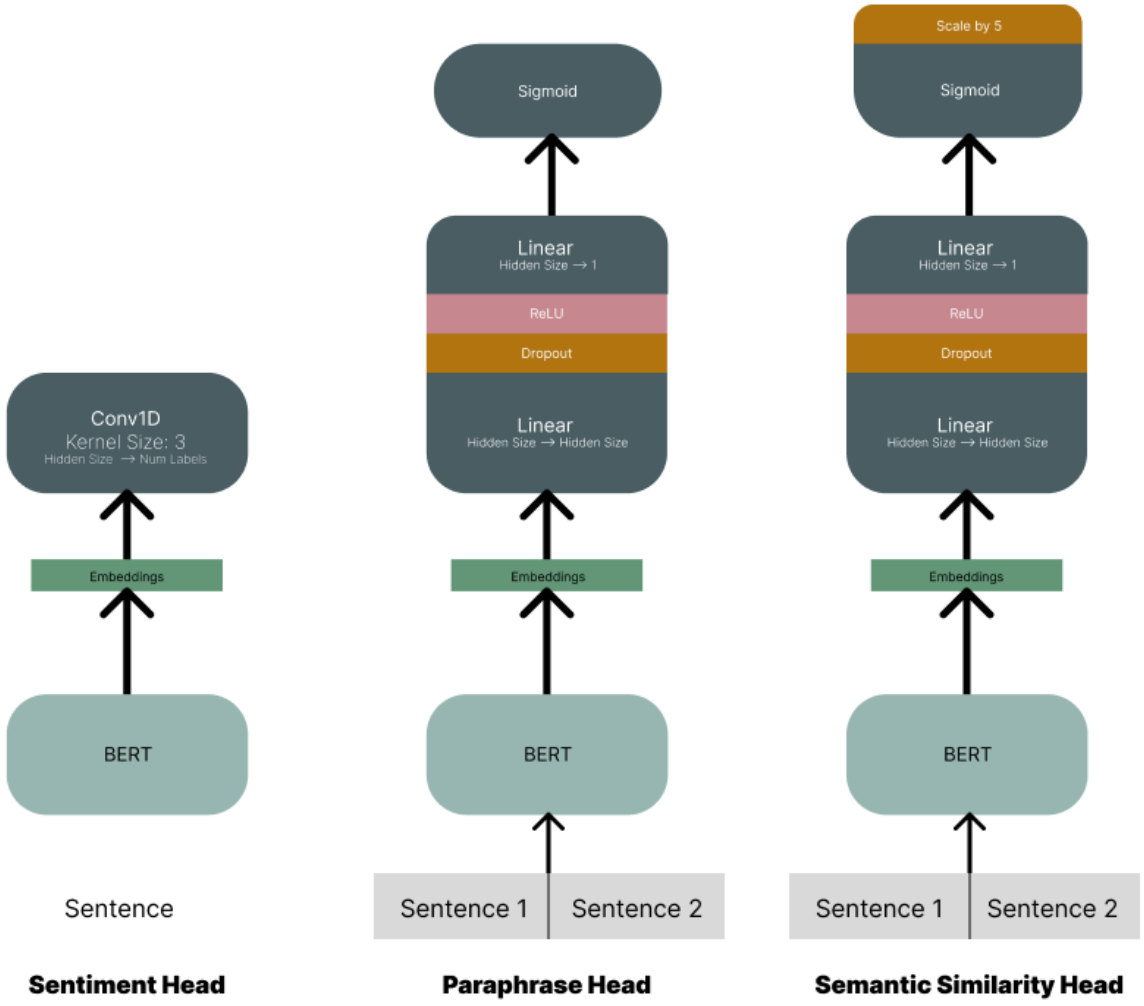

Figure 2: Model Architecture

5.2 Evaluation method

Model performance was evaluated on the following metrics:

1. **Paraphrase Accuracy** on dev set
2. **Sentiment Analysis Accuracy** on dev set
3. **Semantic Text Similarity Accuracy** on dev set

The final metric that was used for overall model performance was the average of these three accuracies. It is worth noting we did have access to a test set, but could only submit three times to see results. Thus, we decided to use the dev set as our main evaluation metric.

5.3 Experimental details

In order to thoroughly test our hypotheses, we trained and tested our model: using the basic heads, the task specific heads, and combinations of data augmentation and gradient surgery. Below are the hyperparameters that were used for each:

- Epochs: 10

- Drop out probability: 0.3
- Hidden Size: 768
- Batch sizes: 8 and 4
- Learning rate: $1 \cdot 10^{-5}$ and $5 \cdot 10^{-6}$

The batch size of our model when using gradient surgery was reduced to 4 due to computation constraints. As a result the learning rate was changed to $5 \cdot 10^{-6}$ as suggested by recent research [8].

5.4 Results

Table 1: Results

	Results			
	SST Score	**Para Score**	**STS Score**	**Avg. Score**
	Baseline Heads			
Normal	0.508	0.709	0.6	0.606
Aug 15%	0.484	0.726	0.478	0.56
Aug 20%	0.51	0.73	0.578	0.606
Aug 40%	0.478	0.747	0.587	0.604
	Task Specific Heads and No Gradient Surgery			
Normal	0.507	0.854	0.859	0.74
Aug 15%	0.499	0.819	0.849	0.72
Aug 20%	0.5	0.856	0.831	0.73
Aug 40%	0.494	0.816	0.856	0.72
Aug 80%	0.818	0.872	0.48	0.72
	Gradient Surgery			
Normal	0.498	0.825	0.857	0.73
Aug 15%	0.5	0.83	0.844	0.72
Aug 20%	0.518	0.834	0.776	0.73
Aug 40%	0.503	0.855	0.832	0.73

Our results show that our carefully designed heads perform much better than our baseline as expected. This makes sense as they had more complex structures, which made use of the expressive power of the BERT embeddings. However, unexpectedly gradient surgery and data augmentation did not improve the performance of our model. This might be due to the limited amount of resources that were available to us. In the case of gradient surgery, we were forced to train the model with a lower batch size and learning rate. Which combined with a limited amount of epochs prevented the model from training fully. In the case of data augmentation, the dataset was relatively well populated and thus augmenting the data did not provide any new insights for the model to learn.

6 Analysis

The best model performance was achieved by modifying the task specific heads. We noticed the largest improvements when we concatenated the inputs and fed those into the BERT model. We believe that this is because BERT is able to form more complex embeddings by having access to both sentences rather than just one. We believe that the concatenation itself did not capture the interactions between sentences completely. Thus, our reasoning was that the additional linear layers, ReLU layer, and dropout layer helped to bridge the gap between the individual sentences. Clearly, this is true at least in part because our model outperformed the non-task specific baseline significantly.

Data augmentation did not significantly improve the model when using 15% and 20%. We believe this is due to how the data is sampled as it is likely that augmented sentences from the same original sentence were over sampled. Furthermore, it is possible that by providing more data that is similar the model overfit, memorizing the data rather than the patterns. Overall, data augmentation, by itself, did not improve or decrease our model significantly. We suspect that if we had limited training data that this technique would have been useful by itself.

In our gradient surgery trials, the train loss convergence was significantly slower than in trials without gradient surgery. We believe that this is in part due to the lowered learning rate and batch size, thus more epochs were necessary to achieve the best parameters.

7 Conclusion

This exploration highlighted the potential for multitask models and their promise for improvement. Primarily, creating task specific model heads allow models to better train for downstream tasks. Secondly, augmenting data on a complete dataset is not as effective as in [4]. This indicates that it still may be useful for future work where data is limited as it did not hinder model performance. Finally, gradient surgery, while an intuitive method for dealing with gradient interference, requires more fine tuning and potentially compute power. All our gradient surgery runs required overnight runs on powerful GPUs which led to a bottleneck of potential runs. Thus, in future work we would like to test the model with more powerful GPUs and larger epoch sizes.
While we improved on our baseline results significantly, there is still room for improvement within our model especially in the Sentiment task. The success of more specific heads for paraphrase and STS suggests that there is a similarly effective technique for sentiment analysis. More complex heads would be another aspect of the project that would be interesting to explore specifically, adding in attention features. Overall, this project served to showcase the potential of multitask models based on an encoder.

8 Individual Contributions

Esteban Cambronero Saba implemented the inital bert model, augmented the data, worked together with Jesus Meza to develop the heads for each of the different tasks, and wrote part of the report.

Jesus Meza worked on integrating gradient surgery, implementing the sentiment head, and wrote part of the report.

References

[1] Jia, Qinjin, Jialin Cui, Yunkai Xiao, Chengyuan Liu, Parvez Rashid, y Edward F. Gehringer. 2021. «ALL-IN-ONE: Multi-Task Learning BERT Models for Evaluating Peer Assessments». arXiv. http://arxiv.org/abs/2110.03895.

[2] Peng, Yifan, Qingyu Chen, y Zhiyong Lu. 2020. «An Empirical Study of Multi-Task Learning on BERT for Biomedical Text Mining». En Proceedings of the 19th SIGBioMed Workshop on Biomedical Language Processing, 205-14. Online: Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.bionlp-1.22.

[3] Crawshaw, Michael. 2020. «Multi-Task Learning with Deep Neural Networks: A Survey». arXiv. http://arxiv.org/abs/2009.09796.

[4] Feng, Steven Y., Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. "A Survey of Data Augmentation Approaches for NLP." arXiv, December 1, 2021. http://arxiv.org/abs/2105.03075.

[5] Yu, Tianhe, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. "Gradient Surgery for Multi-Task Learning." arXiv, December 21, 2020. http://arxiv.org/abs/2001.06782.

[6] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding." arXiv, May 24, 2019. http://arxiv.org/abs/1810.04805.

[7] Tseng, Wei-Cheng. "PyTorch-PCGrad." Python, March 20, 2023. https://github.com/WeiChengTseng/Pytorch-PCGrad.

[8] Goyal, Priya, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. "Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour." arXiv, April 30, 2018. http://arxiv.org/abs/1706.02677.

[9] Parisotto, Emilio, Jimmy Lei Ba, y Ruslan Salakhutdinov. 2016. «Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning». arXiv. http://arxiv.org/abs/1511.06342.

[10] Ouyang, Xi, Pan Zhou, Cheng Hua Li, y Lijun Liu. 2015. «Sentiment Analysis Using Convolutional Neural Network». En 2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing, 2359-64. https://doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.349.