minBERT for Multi-Task Learning

Stanford CS224N Default Project

Maoan Wang
Department of Computer Science
Stanford University
`maoanw@stanford.edu`

Emily Stanford
Department of Computer Science
Stanford University
`emilycs4@stanford.edu`

Abstract

Modern NLP models place a large focus on building robust word embeddings to gain a strong foundation upon which we can build strong performing models. Better embeddings and understanding of the relationships between words would allow us to use the same embeddings across different Natural Language tasks without needing to relearn the relationship between words and phrases. While we can use word embeddings across models, using the same model for multiple tasks can be difficult. In this project, we use Multi-task learning to simultaneously fine-tune a minBert model to perform classification across sentiment analysis, paraphrase detection, and classification of semantic textual similarity tasks. To reuse as much knowledge as possible gained from training of other tasks, we perform backpropagation across each task to support learning across all tasks.

1 Introduction

In this project, we aim to create a classifier that performs multi-task classification across sentiment analysis, paraphrase detection, and classification of semantic textual similarity tasks. These tasks are somewhat related — in our case, paraphrase detection and semantic textual similarity classification are similar tasks that we believe should not require two separate classifiers for classification. Both models would first need to understand and gauge similarity between the input texts before outputting their classification. Instead, it may be more efficient to have one body that calculates similarity and separate heads that then output their respective classifications. In fact, learning representations through one task should additionally improve performance on the other related tasks. Rather than solving a single problem alone, multi-task learning aims to train a model that can solve multiple problems together at once. The model uses layers that are shared across tasks to learn shared representations that are useful amongst all the related tasks. Outputs for separate tasks are then trained on top of these layers to correctly classify the input under the expected specifications. During training, the loss propagates through these shared layers to update the representations, so these layers are constantly being tuned and both tasks gain the benefit from training.

Optimizing for multiple tasks at the same time, however, can prove to be difficult, especially when the tasks are less related or there is conflicting data in the training sets. Often, multi-task learning uses a primary task and an auxiliary tasks to support the learning of the primary task with stronger representations and more data, such as is done in *MTRec: Multi-Task Learning over BERT for News Recommendation*(Bi et al., 2022). In such cases, we can bias the loss function to be more sensitive to gradient updates propagated from the primary task loss over the auxiliary tasks.

2 Related Work

Multi-task learning is a fast-developing field. We focus on the MTRec paper (Bi et al., 2022) which seeks to improve the News recommendation system, where the existing approaches mainly focus on the News titles. Normally pre-trained large models (e.g. BERT) are used to encode the information of titles. However, other useful features like News category and entities are only encoded as shallow features, which is not compatible with the deep BERT encoding. This paper proposed a multi-task learning framework - MTRec - to be able to encode the multi-field information into BERT altogether.Specifically, two auxiliary tasks besides the main task of encoding the News title are constructed. It sets up a category classification to predict the category distribution of News and uses the Named Entity Recognition task to recognize important entities in the title. The loss function then is updated to minimize the loss of the sum of the main task, category classification, and named entity recognition task.

We believe this paper proposes an interesting solution for multi-task learning, which is very similar to the problem we are trying to solve. In our project, we want to apply the BERT model to multiple sub tasks simultaneously, which is similar to the multi-task learning for the News recommendation to predict the title, category and entities all at once.

Besides, because some gradient updates may be contradictory for different tasks, we also employ gradient surgery as defined in the original paper Gradient surgery for multi-task learning(Yu et al., 2020). Gradient surgery projects the gradient of one task onto the normal plane of the gradient of a different task to ensure that the gradients do not conflict during training.

Regarding the hyper-parameter tuning, which is quite important for multi-task learning. One important principle we learned is that the ratio of batch size to learning rate should not too large to achieve a good generalization ability as in (He et al., 2019).

3 Approach

We use a multi-task learning framework to train our model on all three given tasks. With multitask learning, we aim to learn better embeddings across classification tasks for deeper text understanding, such that learning one task will support learning for another.

We are using our minBert model as our baseline. This minBert structure was originally provided to us by the CS224N course staff[1]. We have modified it by implementing multihead self-attention and an Adam Optimizer. More details about the Bert model architecture can be found in the original BERT Paper Devlin et al. (2018).

3.1 New Architecture

We use our pretrained minBERT as our sentence encoder and fine-tune the model for each of our above defined tasks. Each task has a separate head and loss function. We add up the loss for the three tasks before backpropagation and assign the same weight to each loss. We use binary cross entropy loss for the paraphrase detection task and use the MSE loss for the classification of semantic textual similarity task because the label is a real number between 0 and 5. We train each of these heads simultaneously, passing gradient updates from each head to our trainable parameters in the minBERT model, so it receives gradient updates from each task.

On top of the Bert model, we create three separate output heads for each task. For paraphrase detection, we concatenate the two CLS embeddings for the given pair before passing them to the linear layer for final prediction. For the classification of semantic textual similarity(STS) task, we take the average of the hidden states of the last 2 Bert layers, calculate the cosine similarity between the pair, and then apply the ReLU activation function to ensure a positive output before scaling it by 5. For the sentiment classification task, we simply create a new linear projection on the CLS embedding.

3.2 Training Procedure

We train the model by combining the loss of all the three tasks before applying the backpropagation. As the three datasets have different sizes, we tried several ways.

Same batch size. We first tried to use the same batch size for all three datasets, which means for a single epoch, we go through the longest dataset once, but the smaller datasets would need to iterate more than once. It would lead to overfitting quite quickly for the tasks with less data. So we tried only iterating over the smallest dataset once, which would under-fit on the larger dataset since not all

[1]https://github.com/gpoesia/minbert-default-final-project

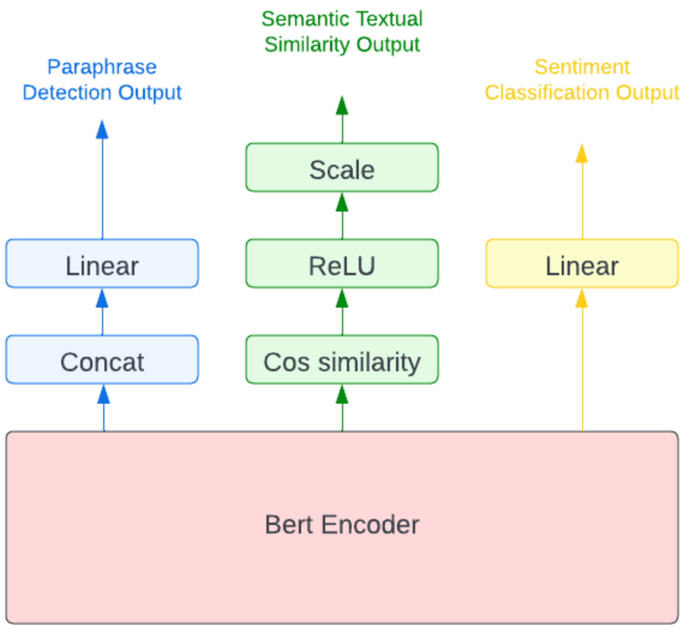
Paraphrase Detection Output
Semantic Textual Similarity Output
Sentiment Classification Output
Scale
Linear
ReLU
Linear
Concat
Cos similarity
Bert Encoder

Figure 1: Diagram of the layers included in the new architecture.

data are used within a single epoch. However, by increasing the number of epochs, we believe all training data would be utilized. This is ultimately the batch strategy that we chose.

Different batch size. We also tried to use different batch sizes to make sure all datasets have been iterated once at the same time for one epoch. However, due to the large difference in size of these datasets, the classification of semantic textual similarity and sentiment classification tasks would only have a batch size of 2, which makes it harder to train these two tasks, even when we upweight the loss accordingly.

3.3 Gradient Surgery

We also applied gradient surgery as in (Yu et al., 2020) to try to tackle the contradictory gradients. We have reused and modified the implementation online[2] though it does not improve the overall prediction accuracy. We will discuss more regarding why it does not work in the analysis section.

4 Experiments

4.1 Data

For sentiment analysis, we use the SST(Socher et al., 2013) and CFIMDB datasets.

For multitask learning, in addition to the SST dataset, we also use the Quora dataset and SemEval STS Benchmark Dataset on the paraphrase detection and text similarity tasks. Evaluation is done only on the dev sets of their respective datasets.

4.2 Evaluation method

We use the average accuracy of all three tasks to measure the performance of the model. We calculate prediction accuracy to evaluate the model on the sentiment analysis and paraphrase detection tasks.

[2]https://github.com/WeiChengTseng/Pytorch-PCGrad

For the classification of semantic textual similarity task, we use Pearson product-moment correlation coefficients.

4.3 Experimental details

Our baseline is our pretrained minBert model trained on the SST and CFIMDB datasets for sentiment analysis without the use of multi-task learning techniques. We trained this model with a batch size of 64 and a learning rate of 1e-3. The model was trained for 10 epochs over 1.5 hours.

We explored different hyper-parameters for fine-tuning. After several rounds of trials, we choose batch size 16 and learning rate 1e-5. The model trained for 10 epochs over 1.5 hours.

We also explored using matrix multiplication instead of a linear projection for the paraphrase detection and the classification of semantic textual similarity tasks, but experienced significant regression in performance.

4.4 Results

The following table summarizes the results of our minBert model after completing the implementation for self-attention and the Adam Optimizer on the SST and CFIMDB datasets.

SST pretrain accuracy	SST finetune accuracy	CFIMDB pretrain accuracy	CFIMDB pretrain accuracy
0.392	0.507	0.763	0.971

Our final fine-tuned model performed with the following results against our baseline:

	SST test Accuracy	Paraphrase test Accuracy	STS test Correlation	Overall test score
Fine-tuned	0.526	0.749	0.823	0.669
Baseline	0.517	0.769	0.318	0.535
Diff	+0.009	-0.02	+0.505	+0.134

It is interesting how our paraphrase test accuracy did not increase over the baseline despite our STS test accuracy increasing significantly with finetuning. This could be related to the batch strategy that we chose, as the Quora dataset used on paraphrasing was very large, and the SemEval STS Benchmark Dataset for semantic textual similarity was the smallest. Choosing this batch strategy likely helped the STS task's evaluation metric the most. Part of our hypothesis was that improvement in one task would help improve the other tasks, but we did not see that effect here.

5 Analysis

We found that the matrix multiplication performed poorly, especially on the STS classification task because the target labels are real numbers between 0 and 5 and it is very hard for the dot products of embedding vectors to be within the given range. By switching to linear projection, it could gradually transform the embeddings to a real value.

	SST dev Accuracy	Paraphrase dev Accuracy	STS dev Correlation	Overall dev score
Matrix Multiplication	0.392	0.376	-0.043	0.242
Linear Projection	0.430	0.693	0.316	0.479

We also found the gradient surgery does not improve the final results though in theory we should see increased prediction accuracy. We did some research and one paper (Shi et al., 2023) just published in ICLR 2023 shows that the gradient surgery used in our approach would not reduce the contradictory gradient as we imagine. It can only slightly reduce the occurrence of conflicting gradients, and in some cases, the gradient surgery even increases it.

6 Conclusion

One thing we learned through the experimentation is that Bert model could be powerful in many NLP related tasks, but it is still a very challenging problem to try to use the same model on multiple tasks at the same time. One important thing is how to design the "heads" based on the Bert model and the corresponding loss functions. For the STS prediction, we tried multiple structures, changed several loss functions but only after we effectively changed the unnormalized predictions to be within 0-5, the prediction accuracy has increased to +80%. Also, how to decide the batch size and learning rate are also important especially when we have multiple datasets with different sizes.

References

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. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Fengxiang He, Tongliang Liu, and Dacheng Tao. 2019. Control batch size and learning rate to generalize well: Theoretical and empirical evidence. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Guangyuan Shi, Qimai Li, Wenlong Zhang, Jiaxin Chen, and Xiao-Ming Wu. 2023. Recon: Reducing conflicting gradients from the root for multi-task learning.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient Surgery for Multi-Task Learning. volume abs/2001.06782.