Multiple Strategies to Improve minBERT Multitask Learning

Stanford CS224N {Default} Project

Colin Kalicki
Department of Biomedical INformatics
Stanford University
ck9898@stanford.edu

Abstract

The Bidirectional Encoder Representations from Transformers or BERT is a transformer-based model that generates contextual word representations [1]. This project has focused on using this model to train and perform across the different tasks: sentiment classification, paraphrase classification, and sentence similarity scoring. A model that could perform this tasks well would be useful as it would not require multiple different models and a demonstration the generalizability of the BERT architecture. Improvement over the baseline implementation came from testing different multitask learning methods and pretraining on in-domain data. Cosine similarity finetuning to improve sentence similarity tasks was also tested but did not show improvement.

1 Key Information to include

- Sharing project: N/A

2 Introduction

BERT has the ability to adapt to new tasks quickly, which makes it an attractive option for multitask learning. This approach involves fine-tuning the pre-trained BERT model on multiple tasks simultaneously. By doing so, the model can leverage the shared knowledge across tasks, which can improve the performance of individual tasks as well as the overall performance of the model.

The success of BERT for multitask learning has been demonstrated in a variety of natural language processing tasks, including text classification, sentence similarity tests, and paraphrase detection. In this project, I first establish a baseline pretrained BERT model with multi-task head. After this, I explore using different methods to improve the performance of pretrained baseline BERT model across all three tasks. These methods include, two algorithms for multitask finetuning, cosine-similarity finetuning for sentence similarity , and pretraining using in-domain datasets. Of these methods, my 'in-turn' multitask finetuning algorithm across all three datasets lead to the largest improvement in overall performance. Pretraining using domain specific datasets also improved the performance of the model across the sentence similarity and paraphrase detection tasks.

The main contributions of this project includes testing between two distinct multitask learning methods and a demonstration of multitask performance improvement using a combination of pretraining and multitask training. Furthermore, I found that implementation fo cosine-similarity for finetuning was not key in improving performance of my model.

3 Related Work

Previous work on improving BERT's performance across multiple tasks has looked at multi-task training algorithms across multiple datasets. Liu et. al [2] looked at training their multitask BERT based model by combining data into a single large dataset set and processing a single batch at a time randomly across different tasks. They showed SotA improvement across many language tasks as their model was able to use information across tasks to improve performance. More detail about their multitask algorithm is in the approach section.

Initial work in the initial BERT construction focused on pretraining the model across a general domain [1]. However, recent work in many different domains have shown that pretraining on domain specific data still improves the performance of these tasks. A specific study by Gururangan et. al [3] showed pretraining in-domain leads to performance gains under both high- and low-resource settings. They showed this in data settings including biomedical and computer science publications, news, and reviews across eight classification tasks.

This work informed my two main strategies to attempt to improve my models performance. Testing different multitask learning algorithms and performing in-domain pretraining.

4 Approach

4.1 Base Model

The base model is the minBERT model with three different finetuning heads for each downstream task. For my baseline model each head includes a linear prediction from BERT sentence embeddings to task specific logits. For the paraphrase prediction and sentence similarity tasks the two different sentence embeddings are concatenated prior to linear prediction. This baseline model was used for intial tests on sentiment classification across the CFIMDB and SST datasets (described below) as an initial testing of the BERT model. And later used/augmented for all other subsequent experiments.

4.2 Strategies for Training on Multiple Datasets

I tested two different strategies for finetuning across different tasks using my base model. The first method I call 'in-turn' finetuning. This method pretrains and finetunes the model using all three datasets using all batches of a given dataset at a time. It performs this training for all three datasets in turn for each epoch (Algorithm 1). The models loss function depends on the dataset that is being trained and the specific task in use (MSE for sentence similarity and cross entropy for the other tasks).

Algorithm 1 Multi-Task Learning: In Turn

```
0: Prepare Data T
0: for t in T do
0:     Pack dataset t into minibatch: D_t
0: end for
0: Shuffle D_t
0: for epoch do
0:     for t in T do
0:         for minibatch_t in D_t do
0:             Compute Loss corresponding to task t
0:             Compute gradient
0:             Update model
0:         end for
0:     end for
```

I compared this method to a similar one I found in the literature and discussed previously [2] The MT DNN method for multitask learning pretrains and finetunes on a single batch for each task at a time with the corresponding loss function. I augmented this method to upsample the data corresponding to SST and STS datasets so their were equivelent batches across all datasets (Algorithm 2). It will thus run for the number of batches corresponding to the largest dataset per epoch.

Algorithm 2 Multi-Task Learning: MT DNN w/ upsampling

```
Prepare Data T
for t in T do
    Pack dataset t into minibatch: D_t
end for
Find largest number of batches D_t
Shuffle each D_t
for epoch do
    for i in largest number of batches do
        Compute Loss corresponding to minibatch_i for task t
        Compute gradient
        Update model
    end for
end for=0
```

These methods are compared to a baseline of just training on the sentiment classification class using the SST data.

4.3 Additional Pretraining on Target Specific Data

The baseline BERT model is trained using wikipedia data (default project handout). I believed that pretraining using domain specific data, or data from my training datasets per each task, would improve my models performance. To perform this each dataset had 15% of the tokens masked. This data pretrains our model by having the masked learning head predict the correct tokens that have been masked. The masked data was fed through the base BERT model with a masked learning head from hugging face using the same pretrained weights as used for our model. Once pretrained, the weights were exported and fed into our model for finetuning using the 'in-turn' method for multi task learning as described above.

4.4 Cosine-Similarity Fine-Tuning

Lastly, I attempted to improve my performance on the sentence similarity task alone by changing my task head from a linear prediction to calculating the cosine similarity between the two embeddings. This value was scale from 0-5 to match the sentence similarity dataset labels. Then MSE loss was computed.

5 Experiments

5.1 Data

To perform initial sentence classification I used the SST dataset and CFIMDB data set [4]. The SST consists of 11,855 single sentences from movie reviews extracted from movie reviews. The dataset includes a total of 215,154 unique phrases each has a label of negative, somewhat negative, neutral, somewhat positive, or positive. The CFIMDB dataset consists of 2,434 movie reviews with a binary label of negative or positive. To pretrain and finetune the sentiment classification head for my multitask model I used the SST dataset. To pretrain and finetune the paraphrase classification head I used a subset of the Quora dataset [5]. It consists of question pairs with labels indicating whether particular instances are paraphrases of one another. To pretrain and finetune the sentence similarity head I used the SemEval STS benchmark dataset [6]. It consists of 8,628 different sentence pairs of similarity rated a scale from 0 (unrelated) to 5 (equivalent meaning).

5.2 Evaluation method

To evaluate classification of both sentiment and paraphrase tasks I used accuracy across the corresponding datasets. To evaluate the sentence similarity task I calculate the Pearson correlation of the true similarity values against the predicted similarity values across.

5.3 Experimental details

Pretraining used a learning rate of 1e-3 and a batch size of 64 and 8 for initial classification of SST and CFIMDB respectively and a batch size of 32 for multihead pretraining. Finetuning used a learning rate of 1e-5 and a batch size of 32. Parameters of the BERT model were frozen for pretraining and unfrozen for finetuning. In-domain pretraining used and epoch of 4 with the same parameters as other experiments (batch size of 32 and pretraining learning rate).

5.4 Results: Strategies for Training on Multiple Datasets

Table 0: Accuracy of single task model prediction on SST and CFIMDB datasets

Dataset	Result
SST	0.511
CFIMDB	0.963

Table 0 shows the initial BERT implementation results from testing on sentiment classification. The results are as expected as SST includes 5 labels while the CFIMDB includes only a binary.

Table 1: Multitask training algorithm results

Task	Baseline Dev	In-Turn Train	In-Turn Dev	MT DNN Train	MT DNN Dev
SST	0.262	0.989	0.478	0.993	0.511
STS corr	0.001	0.987	0.403	0.986	0.395
Paraphrase	0.375	0.933	0.811	0.686	0.522

Table 1 shows that both methods improved over the baseline of just training on the SST dataset (to no surprise). The in-turn algorithm outperformed the MT DNN with upsampling across sentence similarity and paraphrase detection. This result went against my intuition as I expected that training on a batch at a time compared to the entire dataset at a time would allow the model to learn more generalized weights. It could be that the in-turn method in this particular setting (where two datsets have much fewer samples than the largest dataset) is more useful than overfitting, but more experimentation would need to be done to prove this. Nevertheless, both methods lead to an extreme overfitting to the training data as seen in table 1. Interestingly, the overfitting is diminished for MT DNN train to dev sets for the paraphrase detection task.

5.5 Results: Additional Pretraining on Target Specific Data

Table 2: Pretraining Results

Task	W/o pretrain Train	W/o pretrain Dev	W/ pretrain Train	W/ pretrain Dev
SST	0.962	0.478	0.984	0.506
STS corr	0.943	0.403	0.957	0.442
Paraphrase	0.975	0.811	0.971	0.806

Using the previous model trained using the in-turn method as a baseline it becomes clear pretraining on in-domain data improves performance across sentiment classification and sentence similarity analyses (table 2). I was surprised to see a slight decrease in the performance of paraphrase detection. The same extreme overfitting to training data is seen here as well.

5.6 Results: Cosine-Similarity Fine-Tuning

Table 3: Cosine-Similarity Fine-Tuning Results

Task	Linear Pred Dev	Cosine-sim Dev
SST	0.506	0.502
STS corr	0.442	0.403
Paraphrase	0.806	0.807

Cosine-similarity finetuning for the sentence similarity task did not improve my results when compared to using a linear prediction layer. Different implementations of this task, use of the CosineEmbeddingLoss, or aceraging across token embeddings rather than the CLS token could improve

performance, but within the context of the experiment the linear prediction outperformed using cosine similairty.

5.7 Results: Overall

Table 4: Overall Test Set Results

Task	Result
SST	0.514
STS corr	0.393
Paraphrase	0.811

Compared to my best performing model in Table 2 (pretraining with in turn multitask learning) we can see some slight overfitting to the dev dataset.The biggest decrease is with STS performance as that dropped 10 points in the test set suggesting serious overfitting of that multitask head to dev set data. Overall more experimentation needs to be done to reduce the overfitting to training set data. One major experiment that I believe would improve this problem would to get more data for the Sentiment classification and sentence similarity tasks. Nevertheless, the results show that the in-turn multi task learning algorithm combined with in-domain pretraining led to the best performance.

6 Analysis

For intial analysis I focus on my two most promising results. First looking at F1 scores from in-turn learning vs per batch learning. I also look at the F1 scores of Sentiment classification between the pretrained and non-pretrained on target domain data

6.1 Analysis of Strategies for Training on Multiple Datasets

Table 5: Paraphrase Detection F1 scores and Anlysis

Metric	In Turn Dev	MT DNN dev
F1	0.787	0.601

Table 5 shows the stark improvement of F1 score for the In Turn algorithm vs the MT DNN algorithm. This is a more informative metric for accuracy compared to our accruacy metric since around 60% of the dev data for the quora dataset corresponds to 0.

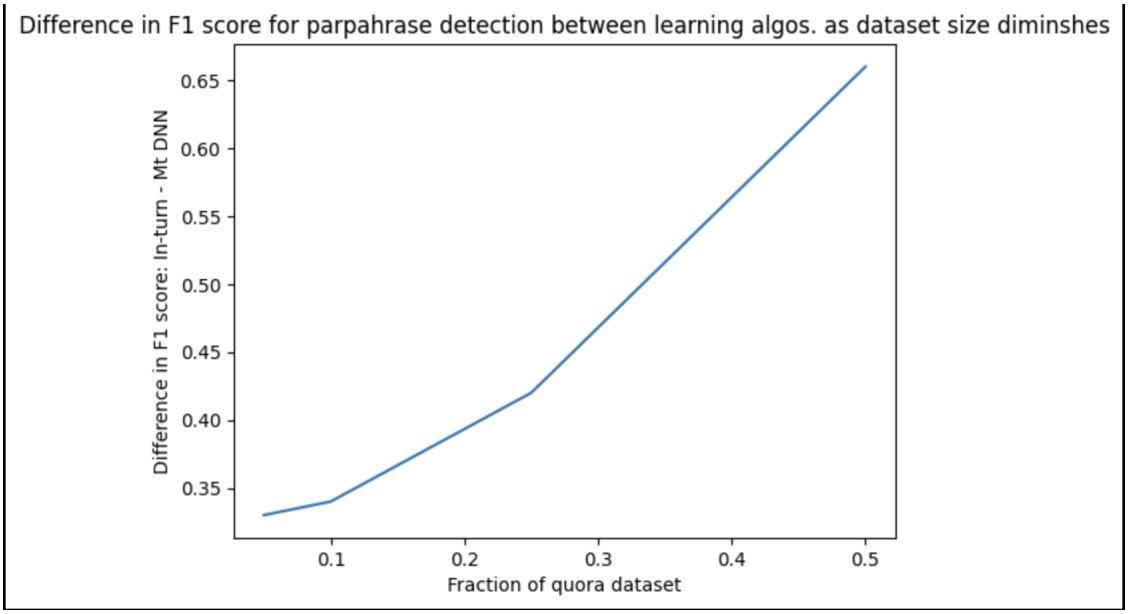

Figure 1: Diminishing difference in performance of multitask learning algorithms as dataset size for paraphrase detection task decreases

Furthermore, in order to test my hypothesis that the performance of the In Turn algorithm was due to the imbalance of data I performed a study diminishing the quora training dataset size and finetuned on 5 epochs (Figure 1). The difference (In Turn - Mt DNN) of the F1 scores of dev performance shows a steep decline as the dataset is reduce to 1/4 of the original size and the performance continues to converge as the datasets are reduced. This possibly suggests that the benefits of the In-turn algorithm are limited to when there is a lot of data in a particular task compared to others. More work will still need to be done to compare the other tasks and different ratio of dataset sizes that are optimal for each multitask learning algorithm.

6.2 Analysis of Additional Pretraining on Target Specific Data

Taking a closer look into the improvement of sentiment classification for each class using F1 scores, table 6 compares in-domain pretraining without in-domain pretraining. It is evident that in-domain pretraining improves not just the overall accuracy but the ability for the model to predict each class. This result is intuitive as it shows that task specific pretraining is effective and the initialization of model parameters with some weighting towards in-domain data improves the model. Interestingly, looking at F1 scores across each class we also see that the model predicts the classes uniformly and seemed to improve uniformly. This suggests that pretraining does not bias the model towards one class but allows the model to learn general parameters effective in improving across all classes. Surprisingly, the best rate is it's ability to predict class 1 at just 0.551. This suggests that a multitask model or BERT architecture, more generally, is not well suited for this task as even after pretraining it struggles to distinguish classes well with low F1 scores even at best.

Table 6: Inspecting Specific Class Performance F1 scores for Sentiment Classification

Model F1 Score	0	1	2	3	4
No Pretrain	0.471	0.513	0.393	0.528	0.430
W/ Pretrain	0.501	0.551	0.405	0.553	0.454

7 Conclusion

The main findings from my project include the the creation of a BERT model backbone for multitask learning and performance across sentiment classification, paraphrase detection, and sentence similarity analysis. I was able to show that my 'in-turn' algorithm for multitask learning is better suited for learning than the MT DNN multitask learning algorithm with up-sampling when there is an uneven amount of data (particularly skewed to a single task). However, this method is not robust across data set sizes as diminshing the size of the large dataset has the methods converge in performance. Furthermore, I was able to demonstrate that pretraining on in-domain datasets does significantly improve the models performance, particularly for the sentiment classification task. Lastly, I showed that cosine similarity task does not improve but worsens the performance of my sentence similarity test. Overall, my final test set performance was 0.811 for paraphrase accuracy, 0.514 sentiment classification accuracy, and 0.393 sentence similarity correlation.

My model's biggest shortcoming was its tendency to overfit on the training set data. This was particularly bad for my in-turn multitask learning algorithm as the train set performance was >0.9 for every task. I unfortunately did not have enough time to try to train on different datasets for each task but strongly believe this would help to alleviate this shortcoming. Furthermore, more work needs to be done on understanding the difference between my in-turn and the mt dnn algorithm. Further ablation studies and varying dataset sizes on the tasks could be done to truly understanding the setting in which each algorithm is strongest.

References

[1] Kenton Lee Jacob Devlin, Ming-Wei Chang and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *arXiv preprint arXiv:1810.04805*, 2018.

[2] Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496. Association for Computational Linguistics, 2019.

[3] Swabha Swayamdipta Kyle Lo Iz Beltagy Doug Downey Noah A. Smith Suchin Gururangan, Ana Marasovic. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 6707–6719. Association for Computational Linguistics, 2020.

[4] Jean Wu Jason Chuang Christopher D Manning Andrew Y Ng Richard Socher, Alex Perelygin and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013.

[5] Samuel Fernando and Mark Stevenson. A semantic similarity approach to paraphrase detection. In *Proceedings of the 11th annual research colloquium of the UK special interest group for computational linguistics*, 2008.

[6] Mona Diab Aitor Gonzalez-Agirre Eneko Agirre, Daniel Cer and Weiwei Guo. Semantic textual similarity. in second joint conference on lexical and computational semantics. In *proceedings of the Main conference and the shared task: semantic textual similarity*, 2013.