Multitask BERT: Exploration and Extension

Stanford CS224N Default Project

Aqil Naeem
Department of Computer Science
Stanford University
aqiln23@stanford.edu

Abstract

This project will be exploring the BERT model and possible extensions. The BERT model has been an important innovation in natural language processing, and performs extremely well on single-task evaluation. The goal of this project is to extend its performance to multitask scenarios, specifically on sentiment analysis, paraphrase detection, and semantic textual similarity. This project explored four ways of extending BERT: including additional dropout layers with layer normalization to reduce overfitting, including RELU layers for performance improvement , including cosine similarity when completing tasks which focus on similarity, and multitask finetuning by training all three tasks at the same time and combining their loss functions. These extensions concretely improved performance of the model over baselines. This project also utilized parameter search to locate ideal hyperparameters for the improved model, which similarly boosted performance on the three tasks of relevance. The main findings were that including additional dropout layers significantly reduced overfitting, cosine similarity was effective in increasing comparison-based scores, and multitask finetuning and RELU layers had smaller, but definite improvements. Increasing batch sizes notably improved efficiency with slight impact on scores, increasing dropout rates decreased overfitting but hurt performance, and increasing the learning rate had harmful effects on the optimal model's performance.

1 Key Information to include

- Mentor: Gabriel Poesia
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

Multitask classification has become a primary focus for model evaluation. While performance on specific tasks yields merit, models are increasingly expected to perform well across a variety of tasks to improve real world performance. Multitask classification is uniquely difficult due to design choices on ensuring equal training for the three methods, conflicting embeddings for certain tasks, and various complexities increased with a multi-task model. While training separate models for individual tasks and then using respective models has been explored, its computationally inefficient and lacks general appeal amongst the NLP community. Therefore, this project aims to adapt the BERT model, which performs extremely well on single-task performance, and explore ways to strengthen performance on the multiple tasks given.The original BERT (Bidirectional Encoder Representations from Transformers) paper published in 2018 outlines the BERT model, relying on transformer architecture. Since then several developments have extended its utility to increase both accuracy and scope of tasks it could work with. Current methods have been successful in improving multitask classification as AI research continues to intensify with the utilization of various novel loss and

training methods, and this paper expects to investigate a unique combination of improvements to understand the marginal benefits of particular design choices. Understanding particular improvements provides further insight into developing more robust systems in the future which will extend natural language processing's role in AI development by understanding which methods show the greatest marginal improvements. Therefore the purpose of this paper is to understand certain methods' impact on performance instead of attempting to build upon state of the art methods to improve accuracy.
When selecting which improvements this paper would focus on, particular importance was assigned to features which would reduce overfitting, improve accuracy, and impact all three tasks instead of singular developments for certain tasks. The motivation behind this paper was to utilize similar methods on all three to deliver superior performance relative to non-improved performance. One exception to this was the inclusion of Cosine Similarity, which impacted two out of the three tasks at hand, but due to its relevancy was an appropiate choice for project design. As a single developer, choices were made with a focus on time-efficient solutions which work within the given architecture. Thorough testing outlines the improvements brought about by these changes and provide significant results which were generally favorable in demonstrating sustained improvement over baseline and non-improved models.

3 Related Work

The aims of this project were heavily influenced by prior work in the field. Although all code for extensions were developed independently, many of the theoretical underpinnings and motivation for particular design choices were inspired by such work.
Inclusion of dropout layers has been shown to be a computationally efficient method for reducing overfitting and improving sustained performance. A primary paper outlining the benefits of this approach, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", outlines how randomly dropping visible and hidden units in neural networks provide improvements, particularly in complex networks with multiple layers, as explored in further extensions developed (Srivastava et al., 2014). The paper shows that models utilizing dropout layers and layer normalization show the greatest improvement in performance relative to other regularization methods (and were therefore utilized in this project). Given that overfitting has continued to be a problem in the field of NLP research, utilizing dropout yields immense benefit due to its effectiveness and computational efficiency.
ReLU, (Rectified Linear Unit) layers, offers a linear-esque activation function to improve performance and capture unique insights on the tasks. As outlined in, "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning", compared to sigmoid functions, ReLU functions are shown to perform empirically at the same or better levels while converging quicker (Nwankpa et al., 2018). This is because sigmoid functions in complex networks will have fractional gradients which when multiplied will lead to exponentially smaller gradients which take longer to converge. ReLU layers instead have gradients which are either 0 or 1, which means multiplying the gradients will not heavily penalize the time to converge.
Cosine Similarity offers an important approach for similarity tasks such as paraphrase detection and the STS tasks provided in this project. The paper "Zhestyatsky at SemEval-2021 Task 2: ReLU over Cosine Similarity for BERT Fine-tuning" provides empirical work demonstrating the effectiveness of a combined ReLU and Cosine Similarity strategy for maximizing performance utilizing BERT (Zhestiankin and Ponomareva, 2021) Relative to other methods, cosine similarity reduces the impact of size within how relative certain tasks are and bounds similarity from 0 to 1, lowering variance. These features outlined in related work make implementing cosine similarity recommendable for similarity tasks, but for tasks such as sentiment analysis the method holds little relevance.
Multitask finetuning entails finetuning the BERT model across all three tasks at the same time during the training loop instead of optimizing for each individual task. Given that the goal of this project is to maximize accuracy across all tasks, combining the loss offers an attractive method of reconciling potentially conflicting learning amongst the tasks. The method utilized in this paper is similar to the approach incorporated in "MTRec: Multi-Task Learning over BERT for News Recommendation", which summed the losses together for an efficient method of minimizing error (Bi et al., 2022) Given that scoring was done based on the average accuracy on the tasks, utilizing a simple sum for loss was an efficient choice. A further exploration in this field not investigated in this paper has been provided in "A Comparison of Loss Weighting Strategies for Multi task Learning in Deep Neural Networks" which explores various methods of augmenting the weighting of certain tasks in the combined loss functions to deliver more robust solutions (Gong et al., 2019).

4 Approach

The architecture of the neural network depends on the various tasks, but were generally advised by relevant literature outlining optimal ordering of layers. Below are descriptions of the extensions implemented and also additional details on critical implementation matters. A diagram on the next page demonstrates the simple architecture utilized. All tasks utilize the forward method which simply returns a dropout layer from the original single-task bert output for a given tasks. All tasks also utilized a cross entropy loss method. Below are descriptions for additional changes based on the task over the baseline and various design choices that held significant impact on the model's results.

Sentiment: Sentiment classification utilizes the following architecture. After taking the dropout layer of the first BERT, it applies a basic linear layer, followed by layer normalization, followed by a dropout layer, and a second linear-layer norm - dropout stack. The final return is an additional linear layer on the final dropout. The network architecture was chosen in line with methods discussed in papers that suggested the following order. The process was repeated twice to include more robust results, as using only a single layer not only had poor performance but had significant overfitting. The inclusion of the dropout layer and layer normalization had great impact.

Paraphrase Each sentence receives the CLS token from forward, and are then concatenated, have a dropout layer applied to them, and are pushed through another linear layer. The similar stack of linear- layer norm - dropout is utilized, but for Paraphrase a ReLU activation layer was utilized after the dropout per the recommendations discussed earlier in the relevant work. This was similarly repeated once more, similar to the sentiment approach, however the final result also included a cosine similarity which was added to the final linear layer, to incorporate both methods.

Similarity The similarity method used the exact same architecture as the Paraphrase task, except its Cosine Similarity was scaled by 5 as Cosine Similarity yields numbers ranging from 0 to 1 and the task involved scores ranging from 0 to 5. The architecture was similar due to how similar the tasks were and to reinforce the original mission of keeping the methods utilized as generalizable across all three tasks. Note that the only difference between sentiment and the similarity tasks are the inclusion of layer normalization, ReLU, and cosine similarity, otherwise the process and repetition of the layers has been consistent across all three predict functions.

Multitask Finetuning: To implement multitask finetuning, a general loss variable was instantiated, and instead of training each task separately, each task was trained within the same training loop and combined towards the end for optimization. The method of combination involved a simple sum due to the final scoring being based on the average of their scores. We can represent this as $\mathcal{L}_{Total} = \mathcal{L}_{STS} + \mathcal{L}_{SST} + \mathcal{L}_{Paraphrase}$.

Training Loop Design: In order to get results which converged, and quickly, the training loop ran batches until the biggest dataset were to be completed. This ensured ample training time. This method was chosen by comparing this method against several others such as taking the minimum size, average size, and increasing the batches for the Quora dataset. In all three instances the maximum size yielded the strongest initial results with quickest convergence. In further work this could be augmented to utilize a random sampler or more novel methods for ensuring equal training and adjusting the number of epochs.

Set Parameters: Certain parameters were not adjusted, most notably the number of epochs. Although adjusting the number of epochs could yield better results, due to the time expansive nature of optimal parameter search, the number of epochs remained at the default value of 10. All other parameters were modified and tested accordingly.

Parameter Exploration: A variety of combinations of parameters were chosen and manipulated. Generally dropout rates were bounded at 0.2 and experimented up to 0.5. The learning rate was mainly adjusted between 1e-5 and 1e-4. Batch sizes were manipulated in multiples of 8 from 8 to 40. Results were completed by running various tasks on the final model with all extensions for the sake of simplicity - certain observations were conducted with manipulating dropout layers with the inclusion of certain layers, however batch size and learning rate were not adjusted dynamically with other extensions developed.

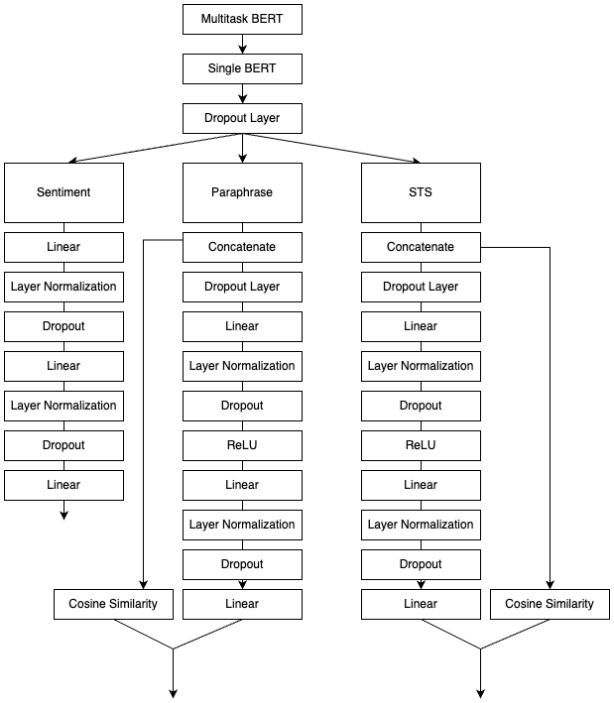

Figure 1: Simple diagram outlining network architecture utilized

5 Experiments

5.1 Data

The datasets utilized in these experiments are the provided Stanford Sentiment Treebank, Quora, and SemEval STS Benchmark datasets. No other datasets were utilized for evaluation or training. All datasets utilized in the quantitative discussion are the test sets. The Stanford Sentiment Treebank dataset contains 11,855 examples with various sentiment labels, Quora contains 400,000 pairs of questions labeled as either paraphrases or not, and SemEval contains 8, 628 sentence pairs with similarity scores from 0 to 5.

5.2 Evaluation method

The evaluation method utilized was simple accuracy as well as Pearson correlation to compare the degree to which sentences are similar. These will be the basis of quantitative measurements. For qualitative methods performance on specific sentences of importance in the dataset and rough time comparisons between batch sizes will be described.

5.3 Experimental details

The model configurations for the experiments used the BERT model with the modifications discussed above. The learning number of epochs was set constant at 10. Parameters which were manipulated include batch size, learning rate, and hidden dropout probability. All tests were run with finetune mode enabled. The models were run on GPUs provided by Google Colab Pro +. No modifications were made to the provided infrastructure for running experiments.

5.4 Results

The final model submitted to the test leaderboard yielded the following scores. This model utilized a batch size of 16 and a hidden dropout rate of 0.4. The learning rate was 1e-5 and epochs were 10.

SST	Paraphrase	STS Correlation	Overall
0.531	0.724	0.569	0.608

For comparison, the results for the project milestone, an appropriate baseline given the lack of extensions, were the following with default parameters:

SST	Paraphrase	STS Correlation	Overall
0.293	0.541	0.048	0.294

As shown, performance more than doubled relative to the baseline, with the greatest increases being within the STS correlation score. These results demonstrate the effectiveness of the strategies described, but also leave significant room for growth for future extensions. Future work may include utilizing other methods, such as Multiple Negatives Ranking Loss, which was not implemented due to time constraints given the variety of models being trained. Below are some results for models with only a select few features that were recorded during the development process.

Implementation of Multitask Finetuning: Multitask Finetuning as a single extension yielded small yet definite improvements for the scores. The scores shown below utilize a batch size of 16 and a hidden dropout rate of 0.4. The learning rate was 1e-5 and epochs were 10.

SST	Paraphrase	STS Correlation	Overall
0.322	0.551	0.095	0.323

These results were lower than expected, which may be due to the simplicity of the implementation utilized. Due to the fact that the Quora dataset loss was not scaled, there could be a possibility that simply summing the losses could've skewed the training process, as the training loop involved finishing the entire dataset during an epoch. As mentioned in the related works, future work could experiment with more advanced methods of multitask finetuning, such as utilizing gradient surgery or weighting losses.

Implementation of Cosine Similarity: Cosine Similarity as a single extension yielded meaningful and definite improvements for the scores. The scores shown below utilize a batch size of 16 and a hidden dropout rate of 0.4. The learning rate was 1e-5 and epochs were 10.

SST	Paraphrase	STS Correlation	Overall
0.331	0.670	0.230	0.407

As shown, the results for implementing just cosine similarity over the baseline showed a roughly 25 percent improvement, with the majority of the improvement being on the tasks involving similarity as cosine similarity was only implemented on these tasks. These results were in line with expectations given the popularity of the method within the existing literature base.

Implementation of Dropout and Layer Normalization: Dropout layers and Layer Normalization were implemented together. The following results demonstrate the impact of utilizing dropout layers over the base model and over the model with cosine similarity implemented. The scores shown below utilize a batch size of 16 and a hidden dropout rate of 0.4. The learning rate was 1e-5 and epochs were 10. The results for dropout and layer norm on the base model are shown:

SST	Paraphrase	STS Correlation	Overall
0.382	0.579	0.124	0.362

The results for dropout layers and layer norm on cosine similarity are shown below.

SST	Paraphrase	STS Correlation	Overall
0.390	0.692	0.345	0.496

As shown, the improvements of using the dropout layers somewhat boosted the base model, but had a much larger impact on the model already implementing cosine similarity. This fits within the papers discussed earlier: dropout layers become increasingly important as models become more complex as

they significantly reduce overfitting. In particular, incorporating two layers of the stacks described in the approach meant that dropout layers were utilized at minimum three times (once in forward, twice per task), allowing for stronger effects. Though not recorded, convergence of the models training accuracies were delayed with higher dropout, but the final epochs showed significantly better results, which fall in line with expectations of how both layers are meant to augment performance .

ReLU Layers: ReLU layers were utilized for the similarity tasks to be computationally efficient. The results of implementing ReLU over the baseline is shown below.

SST	Paraphrase	STS Correlation	Overall
0.297	0.584	0.102	0.328

As shown, the results for utilizing ReLU were decent but not as strong as others. Similar to cosine similarity, these results suggest that utilizing ReLU layers in conjunction with other layers promised stronger performance than just simply including ReLU layers into the baseline model. It was interesting to see that the STS strength was considerably higher than previous, which may be due to slight variation and the fact that the base model had performed so poorly that even minor improvements could yield considerable improvements.

Parameter Search: Changes in batch size had little effect on performance metrics and hold little importance. Increasing the batch size dramatically improved runtime efficiency. Changes in the learning rate had negative impacts on performance, but improved performance at the earliest epochs. The dropout probability had a sweet spot around 0.4. Below is a table outlining the impact of the learning rate on performance using the optimal model with all extensions implemented.

Learning Rate	SST	Paraphrase	STS Correlation	Overall
1e-5	**0.531**	**0.724**	**0.569**	**0.608**
2e-5	0.522	0.725	0.550	0.599
5e-5	0.503	0.715	0.539	0.586
1e-4	0.509	0.709	0.547	0.588

It was surprising to see how higher learning rates harmed performance. The takeaway is that due to the complexity of the model, the learning rate being higher lead to overfitting too early, evident in seemingly lower training loss in early epochs which translated to worse outcomes for later epochs. Additionally, it was interesting to see the variance - for certain tasks performance slightly improved, which may be due to randomness involved in training. This fits within consensus in the literature over how higher learning rates could lead to more instability in results. Lower learning rates were not implemented as several epochs showed drastically low scores, which made testing lower scores not particularly useful. An interesting note was that paraphrase accuracy was less affected by increases in the learning rate, which may suggest that due to the disproportionate size of the Quora dataset, the domination of the dataset of the others during the training loop could have lead to skews towards paraphrase at the expense of the other tasks. Although not recorded, early results of higher learning rates on the baseline model without the complicated stack of layers showed worse performance, which could suggest that the presence of dropout and layer normalization mediated overfitting. Higher learning rates possibly lead to overshooting of maxima, leading to worse results.
The second parameter of main interest was the dropout rate. Manipulating the dropout revealed a sweet spot around 0.4. Below are the results for a series of tests on the final optimal model.

Dropout Rate	SST	Paraphrase	STS Correlation	Overall
0.2	0.501	0.707	0.542	0.583
0.3	0.510	0.714	0.552	0.592
0.35	0.512	0.718	0.559	0.596
0.4	**0.531**	**0.724**	**0.569**	**0.608**
0.5	0.522	0.709	0.554	0.595

As shown, a sweet spot for dropout exists at 0.4. An interesting note was that for earlier epochs, the higher dropouts across the board had much worse results - it was only after finishing all 10 epochs did performance increases start to materialize. The impact of dropout seems strong on the SST

dataset, which could suggest that the approach to SST had more overfitting relative to other tasks. This may be because this was the only task that had two additional dropout layers (not including dropout in forward) while the others had three. Therefore the utility of a higher dropout probability was amplified for that task.

Fundamental Takeaways: The amalgamation of these results suggest that the utilization of these extensions together yield increasing returns and vindicate other papers' findings on the complementary nature of certain extensions on the BERT model. Most notably, the performance of dropout layers when utilized in conjunction with more complex and more layers dramatically improved. As a general theme, the more complex the model, the more important regularization would become. Manipulating batch parameters such as the dropout rate had obvious implications due to the number of dropout layers utilized throughout the architecture. Other parameter changes, like manipulating the learning rate lead to more drastic variation as the extensions compiled together which fits within general consensus amongst the literature. The results of the experiments generally fell within expectations.

6 Analysis

Below are analysis on specific sentences included in the test dataset for each task.

6.1 Paraphrase Task

An example of success for the model includes the following prompt in the Quora Dataset - "Do you like Aaj Tak? Why do you dislike Aaj Tak?" The model correctly labeled this as not a paraphrase, which demonstrates that the model successfully overcame basic errors that models can fall into due to the the similar word count and the same words. By being able to distinguish between like and dislike despite having the same words, the model shows complex understanding beyond just counting the number literal words shared between the sentences.

An example of failure for the model includes the following prompt in the Quora Dataset - "What causes narcissism? Is narcissism a mental illness or a personality trait?" Despite the sentences being distinct, the model labeled the sentence as a paraphrase. A possible reason could be the large amount of instances where identifying the cause of a sentence could lead to two options i.e. "Who broke this, Brandon or Allen? Did Brandon or Allen break this?". In this instance the model failed to distinguish the cause of something from its label, which represents a unique example amongst the dataset. Including examples of this particular quirk in the English language could improve future performance.

6.2 SST Task

An example of success for the model includes the following prompt in the SST Dataset - " Ana is a vivid vibrant individual and the movie 's focus upon her makes it successful and accessible ." which was labeled as 4. This shows that the model correctly understands positive words, and understands how repeating positive adjectives signifies strong sentiment.

An example of failure for the model includes the following prompt in the SST Dataset - "This is what IMAX was made for : Strap on a pair of 3-D goggles shut out the real world and take a vicarious voyage to the last frontier – space ." which was labeled a 2 or neutral. This reflects that the model often relies on the presence of certain adjectives, but when presented with words that form positive opinions without explicitly positive words, the model tends to fail. This reflects the reliance on certain words to signify the mood of a sentence, and could be alleviated through including similar words in the training set.

6.3 STS Task

An example of success for the model includes the following prompt in the STS Dataset - "A band is performing on a stage. A band is playing and singing on a stage." which was rated a 4.7 by the

model. This reflects that the model understands what performing entails, which reflects the model's sophistication by understanding more than just literal similarity.

An example of failure for the model includes the following prompt in the STS Dataset - "You should do it. You should never do it." A simple example, the model labeled this a 4.35, when in reality it should be labeled towards a 2 or 1. This shows that the model tends to value the number of words that are similar over certain words that may have significance - in this instance, the model failed to learn the heavy weight the word "never" has in the English language, which would lead to a much lower STS result.

7 Conclusion

This project explored possible extensions to optimize BERT's performance on multiple tasks. The findings of this project show that the described extensions successfully increased the performance of the model, with each extension having independent impact and stronger confluence between features. The inclusion of cosine similarity dramatically improved performance for comparison-based tasks, while the utilization of dropout layers and normalization ensured that as the model incorporated more complex layers the model refrained from overfitting to the training data. Other changes such as utilizing ReLU layers and summing losses on the three tasks showed boosts in scores but to lesser extents when implemented independently. The parameter search with the final model identified unique aspects on how the model interacts with various parameters - for example, the learning rate had a negative effect on accuracy while increasing dropout rates had a stronger increase to the results of the project. A major insight from the project has been that dropout layers become increasingly important as a model becomes more complex - the more layers there are, the higher risk of overfitting which makes maintaining appropriate amounts of dropout layers essential.

There have been limitations to this work. For one, multitask finetuning could have utilized a more sophisticated approach, and various training methods would have yielded different impacts on the performance. Changing the way the model was trained could yield different results on the extensions and could possibly boost performance. As a single developer, there were time constraints associated with implementing certain extensions which precluded the inclusion of certain extensions such as Multiple Negatives Ranking Loss. Future work could include implementing other extensions, including gradient surgery for multitask finetuning, and testing out various training methods to examine performance results. Nonetheless, the results of the project affirmed many of the original takeaways from related work and demonstrate the effectiveness of several methods on improving multitask accuracy.

References

Qiwei Bi, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Hanfang Yang. 2022. Multi-task learning over bert for news recommendation. In *In Findings of the Association for Computational Linguistics: ACL*, pages 2663–2669. Association for Computational Linguistics.

Ting Gong, Tyler Lee, Cory Stephenson, Venkata Renduchintala, Suchismita Padhy, Anthony Ndirango, Gokce Keskin, and Oguz H. Elibol. 2019. A comparison of loss weighting strategies for multi task learning in deep neural networks. volume 7, pages 141627–141632. IEEE Access.

Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. 2018. Activation functions: Comparison of trends in practice and research for deep learning.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research.

Boris Zhestiankin and Maria Ponomareva. 2021. Zhestyatsky at SemEval-2021 task 2: ReLU over cosine similarity for BERT fine-tuning. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 163–168, Online. Association for Computational Linguistics.