A Rigorous Analysis on Bert's Language Capabilities

Stanford CS224N Default Project

Gaurav Rane
Department of Computer Science
Stanford University
grane@stanford.edu

Abstract

Text classification is a major task of natural language processing, and models need to be able to categorize the meaning of different texts. GPT from OpenAI, ULM-FiT from Howard and Ruder, and BERT all attempt to solve that issue of text classification. Our approach focuses on BERT in order due to its mutability and adaptability for numerous textual analysis tasks. Our BERT will perform sentiment analysis, paraphrase detection, and semantic textual similarity. To make a robust Bert that is capable of these tasks, we must pre-train a base implementation on two sentiment datasets: Stanford Sentiment Treebank and CFIMDB. We test the performance of the base implementation of BERT on these datasets, then we further train the model to perform on the three tasks. The pre-training of unlabeled data and multi-task learning are chosen because they are both related approaches that have been proven to improve model accuracy. We find that multi-task learning through additional layers that specialize for the given task can improve model performance for specific tasks, with multi-task learning including the addition of linear layers, cosine similarity tests on similarity tasks, and further fine-tuning so the model gets exposure to the tasks at hand.

1 Key Information to include

- Mentor: None
- External Collaborators (if you have any): None
- Sharing project: No

2 Introduction

The pandemic was heavily dominated by a surge in blockchain technologies, with an unprecedented number of funding allocated towards companies and start-ups working in the crypto space. However, the emergence of GPT-3.5 marked a heavy shift toward a focus in AI, specifically, companies working on natural language technologies. Natural language and deep learning isn't a particularly new concept though, but the power of GPUs and their ability accelerate the training process of billion parameter large language models has allowed for the proliferation of LLM models that can actually produce human interpretable responses Hong et al. (2024).

GPUs accelerate LLM training for performances over all tasks, but our approach will focus specifically on text classification. Text classification is a major task of natural language processing, and models need to be able to categorize the meaning of different texts. GPT from OpenAI, ULM-FiT from Howard and Ruder, and BERT all attempt to solve that issue of text classification Sun et al. (2020). However, due to the diverse nature of text classification tasks, it becomes challenging for a model to be able to adapt and learn for all the different classification methods we see in daily use of language. Whether it be determining the mood of a phrase based on the sentiment of words in the

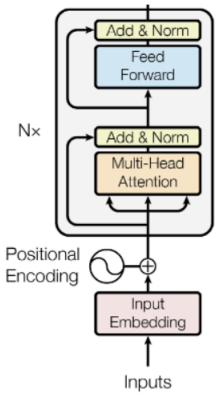
statement, or assessing similarity of two different inputs, language has many more intricacies beyond just general grammar and syntax rules.

Thus, our paper explores the ability of a large language model to perform multi-task learning on three different classification assessments: sentiment analysis, paraphrase detection, and semantic textual similarity. The model must learn different features for each task, and additionally know when to apply the specific learning trait for its respective dataset. To make sure our model learns what's required for the specific task, we plan on using multiple approaches, from further fine tuning for the new tasks and introducing a cosine similarity metric to get more accurate representations of similarities between sentences.

3 Related Work

3.1 BERT

For starters, it's imperative to understand the BERT model that we are working with. BERT is a pre-trained model Devlin et al. (2018) that utilizes the attention mechanism and transformer structure Vaswani et al. (2017). Base BERT utilizes 12 Encoder Transformer layers, where each transformer layer uses multi-headed attention, followed by a sandwiched feed-forward layer between two additive and normalization layer with a residual connection.



This bidirectional approach is key to understand the context of a word based on both its left and right surroundings, allowing for the proliferation of LLM's ability to perform in various text classification tasks.

3.2 Layer Learning Rate Decay

In general LLM architecture, pre-training prepares the model for general language understanding, and fine-tuning with specific metrics is the key to getting a language model to perform for specific task-based functions. Pre-training is expensive though because it has to be retrained for each specific task. However, this multi-task learning (MTL) with BERT is no problem since it uses the same pre-trained model Sun et al. (2020) for every task . Additionally, the paper introduces a novel idea with regards to fine tuning, in which a varying layer rate is used for each different layer according to the below equation.

$$\theta_t^l = \theta_{t-1}^l - \eta^l \cdot \nabla_{\theta^l} J(\theta),$$

A base learning rate η^L is set and each subsequent layer is updated as $\eta^{k-1} = \xi \cdot \eta^k$. ξ is a decay factor less than or equal to 1, and the idea is to figure out just how much of the previous layer's knowledge we should consider.

3.3 Cosine Similarity

As shown with the decay learning rate, fine-tuning involves specific approaches depending on the task, so for tasks involving comparison of similarity between two sentences, another creative approach as shown by Niels Reimer and Iryna Gurevych is to use a cosine similarity test Reimers and Gurevych (2019). Intuitively, the cosine similarity test serves as a metric for how similarly pointing are the directions of each sentence embedding The study conludes that cosine similarity is able to reduce computation time on several hours of magnitude (65 hours to 5 seconds) while also maintaining accuracy with the original model.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

4 Approach

4.1 Dataset

In total, four datasets are used, with two being used to pretrain the base BERT, and then one of those datasets is used along the two unused datasets for the three downstream tasks. For the pretraining step, we load a minBERT model and then prepare it for sentiment analysis on the Stanford Sentiment Treebank (SST) dataset Socher et al. (2013) and the CFIMDB dataset. SST is sentiment analysis over single sentence movie reviews with 5 categories annotated by human judges, and CFIMDB is a collection of movie reviews with binary labels of positive or negative sentiment.

Then, for the three downstream tasks, SST is used again for sentiment analysis, a Quora dataset that is used for paraphrase detection Quora (2017), and semantic textual similarity (STS) is assessed through a dataset from a paper that coined the idea of an STS assessment for LLM's Agirre et al. (2013). The Quora dataset was released by Quora in order to make sure that there is only a single question page for each logically distinct question. Both the Quora dataset and the STS dataset deal with semantic equivalence between two inputs, but Quora is assessed on a binary label because the inputs can either be merged into a single thread on the website or they are logically different. STS, on the other hand, crowdsourced annotations to identify the similarity of two inputs on a continuous scale from 0 to 5.

4.2 Base BERT

To begin our text classification journey, we begin by preparing our BERT base model and retrieving the scores for its performance on the two sentiment analysis datasets. Specifically, attention is computed as a dot product of queries and key vectors, then we divide the value by the square root of dimensions before multiplying by a value vector and taking its softmax.

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Ultimately, each block of the transformer architecture consists of a multi-head attention layer, an add-norm layer that applies dropout to the output of the previous multi-head attention layer and adds the previous input, a feed forward layer, and finally an add-norm operation taking the input and output of the feedforward layer.

Learning for the model is accomplished through an Adams optimizer with weight decay, and then we pretrain and finetune on the SST and CFIMDB dataset. Pretraining is done with a learning rate of $1e-3$ and fine-tuning is performed with a learning rate of $1e-5$. After the base BERT is implemented, and pretraining and finetuning have finished, we received the following scores on the SST and CFIMDB dataset.

Dataset	Score Type	Dev Set Score
SST (Sentiment)	Accuracy	Pretrain: 00.399521
SST (Sentiment)	Accuracy	Finetune: 0.521
CFIMDB (Sentiment)	Accuracy	Pretrain: 0.796
CFIMDB (Sentiment)	Accuracy	Finetune: 0.967

4.3 Multitask Baseline

Now, the multi-task learning comes in, and different specifications are applied to the model depending on the task at hand. For sentiment, we add an additional learning layer for the five sentiment specifications. For paraphrase detection and semantic textual similarity, we add an additional layer that is formed by the concatenation of the absolute value difference and element wise multiplication of the two input tensors. Pretraining and finetuning are performed at the same learning rate specified for the base BERT implementation.

For the baseline scores on the three downstream tasks, we designed our approach as follows. Since our base BERT was pre-trained only on sentiment analysis, our baseline scores will then only be fine-tuned on the SST dataset using a cross entropy loss, due to its categorical label distribution. The additional layers added to the paraphrase similarity task and semantic textual similarity task, as mentioned above, serve as the metric for baseline assessment. The operation of taking the difference is meant to map some sort of a correlation between the two inputs who's magnitude can signify the variance between the two, and the multiplication is meant to magnify both similarites and differences in embedding scores. After fine tuning our model with these baselines set, we received the following scores.

Dataset	Score Type	Dev Set Score
SST (Sentiment)	Accuracy	0.527
Quora (Paraphrase)	Accuracy	0.393
STS (Textual Similarity)	Pearson	0.120

To improve upon these baselines, we plan on adopting several fine-tuning strategies. We will fine tune on the Quora and STS dataset to make sure our model learns paraphrase detecting and semantic textual similarity before being tested on it, and additionally we will utilize cosine similarity for each respective task. The specifications of each model and their accuracy on the dev and test sets is stated in the subsequent section.

5 Experiments

5.1 Evaluation method

Given the categorical distribution of the labels for the SST datset, we will calculate our model performance using accuracy against the dev and test set, with the use of cross entropy loss function during training. For the paraphrase detection task, the Quora dataset has binary labels, so we will use accuracy for model performance as well, but this time use binary cross entropy loss during training. Finally, for the STS dataset, the continuous distribution of labels makes it hard to get exactly the same value predicted from our model, so we will use Pearson correlation coefficient for assessment and means squared error (MSE) loss during training. Due to the fact that the model was already pretrained on sentiment analysis and further finetuned, no additional tests were attempted to increase model performance on the SST dataset.

5.2 Additional Finetuning

The experiment begins with configuring a very rudimentary version of the model and then collecting baseline assessment statistics, which have been reported earlier in the paper. From there, the next step is to actually fine tune on the different tasks we have. Pretraining made the model learn the language, then we fine tuned on sentiment analysis but we still have paraphrase detection and semantic textual similarity to configure. Two of the three baseline statistics were measured on a model that had never seen an instance of that task - zero-shot learning. We hope that by incorporating the Quora and STS dataset in the training process during finetuning, our model will adjust its weights to actually learn for those tasks.

Due to the vast size of the Quora dataset, we randomly chose a tenth of the training examples for each epoch of training. All learning for any portion of the finetuning is executed with a learning rate of $1e-5$, and the losses for each training set are explained in the preceding subsection. Once trained, we then calculate the scores of the model against the dev set for the three tasks. Since both accuracy and Pearson correlation coefficient occur on a scale of zero to one, no further scaling is necessary and we save the model that performs the best on aggregate performance over all three tasks.

5.3 Cosine Similarity on Similarity Tasks

Both the paraphrase detection and the semantic similarity task involve a comparison of two input embeddings, and though the metric of assessment is binary for the paraphrase and continuous categorical for the semantic similarity, both tasks can be boiled down to a simple similarity assessment. Thus, we will implement a cosine similarity metric (see related works for equation) on both the Quora and STS dataset during training. For the paraphrase detection task, the logit gets scaled by a sigmoid, but for the STS there is a bit more room to work with. Since a cosine similarity score outputs a value from -1 to 1, and we need to ensure our outputted logit is a probability, we will test model performance on the STS dataset with sigmoid and relu scaling.

Paraphrase results - baseline with embedding difference and multiplication - same input embedding operation but also add the finetuning on the dataset - operation performs really well, so keep and add cosine on top of baseline instead of overhauling basline operation

STS results - baseline with embedding difference and multiplication (then cancel this approach and do cosine similarity only) - no training data w/ cosine + sigmoid (cosine really improves so keep this over baseline) - training data w/ cosine + sigmoig - training data w/ cosine + relu

Report how you ran your experiments (e.g., model configurations, learning rate, training time, etc.)

5.4 Results

On the STS dataset, we attempted four major training tune ups to ensure a robust model performance. First, we used the baseline approach of embedding difference then embedding multiplication. Then, we got rid of that and solely used cosine with sigmoid, then added the training data to that approach, and finally we finished with a model finetuned on STS training data using a cosine similarity layer with ReLU activation.

Dataset	Model Additions	Dev Set Score
STS	Baseline	0.120
STS	Cosine + Sigmoid	0.446
STS	Cosine + Sigmoid + Train Data	0.473
STS	Cosine + ReLU + Train Data	0.532

We see that the cosine similarity significantly improved model performance, and the training data only performed a slight boost. At the end though, the cosine similarity with ReLU activation and

training data was the optimal fine-tuned approach for STS.

On the Quora dataset, we attempted three major markups to improve the model performance on the dev and test set. First was the baseline with embedding difference and multiplication. Then, we added training data, and since the model performed well with the baseline layers and training data, we added cosine similarity on top of that, instead of removing the baseline layers and replacing with cosine similarity like for the STS dataset.

Dataset	Model Additions	Dev Set Score
Quora	Baseline	0.393
Quora	Baseline + Train Data	0.825
Quora	Baseline + Train Data + Cosine	0.841

Each additional markup to the finetuning process just kept improving the score of the paraphrase dataset, and the combination of everything yielded the best dev set score.

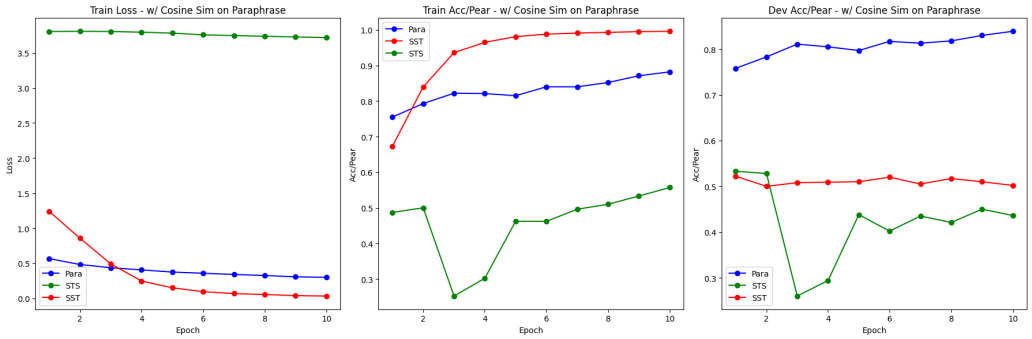
We identified the optimal performance on STS was with training data and cosine similarity with ReLU activation. For the Quora dataset it was keeping the baseline architecture but also adding a cosine similarity score and training data. Two submissions to the test set were done, where the first was with all the optimal adjustments mentioned minus the cosine similarity component on paraphrase detection.

Submission 1		Submission 2	
Dataset	Test Set Score	Dataset	Test Set Score
SST	0.500	SST	0.517
Quora	0.824	Quora	0.839
STS	0.452	STS	0.498
Overall	0.683	Overall	0.702

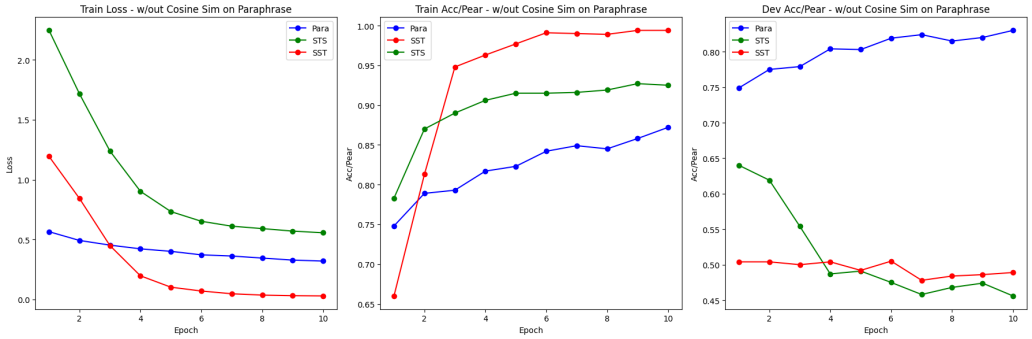
Based on all the training and individual testing, STS is the only dataset with a noticeable lower value on test than the dev set, but still the second submission with the optimal model settings was able to achieve a very high overall score. Unfortunately, the score is such that paraphrase detection dominates accuracy so the overall of 0.702 is heavily carried by one of the scores as opposed to being representative of model performance over all three tasks holistically.

6 Analysis

Overall, the output for the test set shows us that multi-task learning is very much possible, but at a cost. If we inspect the individual breakdowns and not just the overall score, we see that the paraphrase detection is amazing, but both SST and the STS datasets don't have the highest test set scores. Several reasons for this include the datasets for SST and STS are just harder to predict due to their categorical nature while the Quora dataset is purely binary. Additionally, the sheer size of the Quora dataset is magnitudes larger than the others, so perhaps the finetuning pushed the model a little too much to overfitting paraphrase detection and not the other tasks. However, more information is shed on this when we consider what the graphs look like during training.



This above graphs represent the optimal fine tuning configurations minus the cosine similarity for the paraphrase detection task. With regards to the training, STS has some really abnormal graphs for the training accuracy and dev set accuracy. Both STS and paraphrase detection involve identifying similar inputs, so if the paraphrase detection is told to identify similarity one way (without cosine) and STS is told to use another method, then the model will struggle to generalize the concept of a similarity to both tasks and overfit one. This hypothesis makes sense because when we do incorporate cosine similarity in the paraphrase detection, we see the following.



On the training graphs, the problem seems to vanish as now the model learns to use cosine similarity for both paraphrase detection and STS, so the curves follow a much more expected pattern throughout the epochs. The dev set, on the other hand, is a little over the place for paraphrase and STS, suggesting that the model might be overfitting for the parphrase detection task.

7 Conclusion

Our paper identified a way to pretrain and finetune BERT to perform on not just a single text classification task, but three. We found that cosine similarity is crucial for tasks comparing two different inputs. If there are multiple two input based tasks, and we only used cosine similarity for one and not the other, the model doesn't truly understand how to use cosine similarity in the context of comparison, so it's important to standardize the approach across all the tasks that could possible benefit from it.

Finally, when we look back to just the overarching question of the feasibility of multi-task classification, we found that it was quite difficult to achieve high scores over all the tasks. The model usually learns one task really well and the other ones not as much. We were able to bring up the scores of the tasks that the model was not performing too well on, but the biggest limitation was how our overall task score was 0.702 but it wasn't reflective of individual test set performance.

References

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational*

*Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA. Association for Computational Linguistics.

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.

Ke Hong, Guohao Dai, Jiaming Xu, Qiuli Mao, Xiuhong Li, Jun Liu, Kangdi Chen, Yuhan Dong, and Yu Wang. 2024. Flashdecoding++: Faster large language model inference on gpus.

Quora. 2017. First quora dataset release: Question pairs. `https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs`. Accessed: date.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

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.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2020. How to fine-tune bert for text classification?

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.