Multi-Task Learning BERT Model with Task-Specific Decoders

Stanford CS224N Default Project

Zhen Li
Stanford University
zhenlili@stanford.edu

Abstract

In this project, we developed a BERT-based multi-task learning model with task-specific decoders for three downstream NLP tasks: Sentiment Analysis, Paraphrase Detection, and Semantic Textual Similarity. The proposed model includes a BERT-based encoder shared between these three tasks to understand the semantic meaning of the input, and 3 separate decoders composed of self-attention and linear projection to transform the understanding into task-specific predictions.

Main contribution includes: Come up and develop task-specific decoder with self-attention layer. Separate loss backpropogation on each sub-task. Separate optimizer configurations(lr, weight_decay) for each sub-task are used to overcome different convergence speed and training data imbalances problems.
On test leaderboard, our best model achieves **0.492** on Sentiment Classification Acc, **0.867** on Paraphrase Detection Acc, **0.867** Semantic Textual Similarity Corr, the overall test score of **0.742**.

1 Key Information to include

- Mentor: Tathagat Verma
- Professor: Chris Manning

2 Introduction

Understanding and processing natural language is a fundamental problem in artificial intelligence, with a lot of applications in sentiment analysis, paraphrase detection, and semantic textual similarity. The growing demand for robust, efficient, and versatile models that can handle multiple tasks effectively has led to the development of multi-task learning models, which improve performance by leveraging shared knowledge between tasks (Caruana, 1997). Recent advances in deep learning, particularly pre-trained language models such as BERT (Devlin et al., 2018), have achieved state-of-the-art performance on various NLP tasks.

The main challenges in multi-task learning for NLP tasks lie in finding a common representation that can capture the unique features of each task while still allowing for shared learning across tasks. Moreover, it is essential to address the trade-offs between model complexity, training efficiency, and performance on individual tasks.

In this project, we proposed a multi-task learning model by using a pre-trained BERT encoder with task-specific decoders for sentiment analysis, paraphrase detection, and semantic textual similarity. The key idea behind our approach is to leverage the pre-trained BERT model as a shared encoder to learn a common representation of the input text and then employ task-specific decoders to generate task-specific outputs. By incorporating task-specific decoders, it allows for a more nuanced modeling of each task's unique characteristics and enables the model to perform well across different tasks.

We also introduced an self-attention layer in our decoder to help the model learn the optimized pooling for each specific tasks. Besides all above, a lot of effort have been put into balance the performance between different tasks due to the imbalanced training datasets. We came up with an approach to use separate loss backpropogation on each sub-task during each training batch. This allows us to tune separate optimizer configurations(lr, weight_decay) for each sub-task to make them converge at the same pace.

3 Related Work

3.1 Multi-Task Learning

Multi-task learning (MTL) is an effective approach to improve model performance by learning multiple related tasks simultaneously, exploiting the shared information among tasks (Caruana, 1997). MTL has been successfully applied to a variety of NLP tasks, such as machine translation (Dong et al., 2015), and part-of-speech tagging (Søgaard and Goldberg, 2016).

3.2 Pre-trained Language Models

Pre-trained language models, such as BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019b), have achieved state-of-the-art performance in various NLP tasks by leveraging large-scale unsupervised pre-training followed by task-specific fine-tuning. Both of these models use bidirectional transformer architectures (Vaswani et al., 2017), which enable it to capture both syntactic and semantic information from the input text.

3.3 Multi-Task Learning with Pre-trained Language Models

Several works have investigated combining of multi-task learning and pre-trained language models to improve performance on multiple NLP tasks. Liu et al. (2019a) proposed the Multi-Task Deep Neural Network (MT-DNN), which uses BERT as a shared encoder and employs task-specific output layers for various tasks.

3.4 Sentiment Analysis, Paraphrase Detection, and Semantic Textual Similarity

Sentiment analysis aims to determine the polarity (positive, negative, or neutral) of a given text, while paraphrase detection is concerned with identifying whether two text fragments convey the same meaning. Semantic textual similarity measures the degree of similarity between two text fragments in terms of their semantic meaning. These tasks have been widely studied in NLP, with various methods being proposed, including traditional feature engineering techniques (Pang et al. (2002); Dolan et al. (2004)), deep learning models (Socher et al., 2013), and pre-trained language models (Devlin et al. (2018); Reimers and Gurevych (2019)).

4 Approach

4.1 Model Architecture

The proposed model architecture is shown in Figure 1. It includes the following main components:

4.1.1 Pre-trained BERT Encoder:

The pre-trained BERT model we use is $BERT_{Base}$ which has 12 layers, 768 hidden layer dimensions, 12 heads, and 110M total parameters. It's pre-trained on BooksCorpus (800M words) and English Wikipedia (2,500M words). Two main pre-trained tasks are "Masked language modeling" and "Next sentence prediction". (Devlin et al., 2018)

4.1.2 Task-specific Decoder:

Each task has its own decoder which includes an attention layer, followed by a linear projection layer and prediction head. Task-specific layers do not share parameter weights. During training time, each sub-task loss runs backpropagation separately.

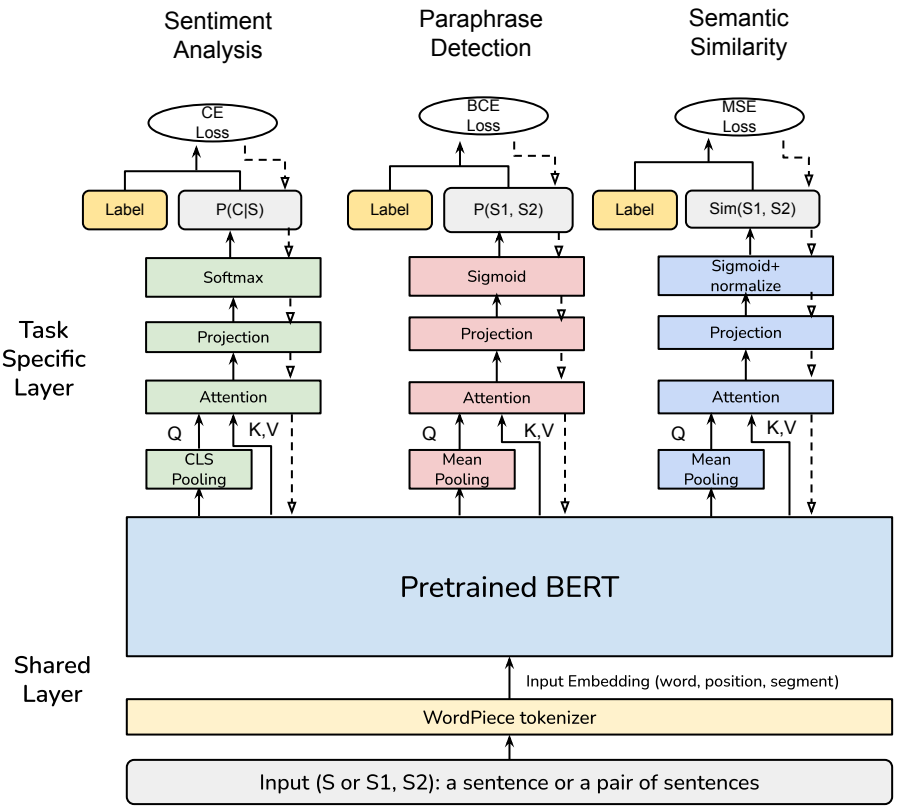

Figure 1: Model Architecture Diagram

4.1.3 Attention Layer in Decoder:

It's worth to mention that we added a Attention Layer between BERT encoder's output and Linear Layer. Its Query input is from the BERT pooling output, the Key and Value is from BERT output hidden states. This is because that during early phrase of project different pooling methods were experimented on each task and it had noticeable impact on the quality. So the idea is that by adding this Attention Layer it can learn the optimized pooling from on the hidden state for each sub-task.

4.2 The Training Procedure

4.2.1 Training Algorithm

The training procedure runs as shown in Alrogithm 1. Note that each task has its own sample_size, batch_size and optimizer_config which can be configured and tuned. The loss function for different tasks are discussed in 4.2.2.

4.2.2 Loss Function

Sentiment Analysis task uses Cross-Entropy Loss as the loss function. M is number of classes, y is binary indicator (0 or 1) if class label c is the correct classification for observation o, p is predicted probability observation o is of class c. Loss defined as:

$$L(\Theta) = -\sum_{c=1}^{M} y_{o,c} \log(p_{o,c}) \tag{1}$$

$D_1, D_2, D_3 \leftarrow load\_data()$
$epoch_{max} \leftarrow args.epoch$
for *epoch in {1, 2, ..., epoch*$_{max}$*}* **do**
  for *t in {1, 2, 3}* **do**
    $S_t \leftarrow random\_sample(D_t, args.sample\_size_t)$
    $S_t \leftarrow random\_shuffle(S_t)$
  end
  # each task has its own sample size and batch size
  # but number are aligned to make sure sample_size / batch_size are equal across all tasks
  $batch\_num \leftarrow sample\_size/batch\_size$
  for *i in {1, 2, ..., batch_num}* **do**
    # inside each mini-batch compute loss and backpropagate on each task
    for *t in {1, 2, 3}* **do**
      $b_t \leftarrow read\_next\_batch(S_t)$
      $optimizer_t \leftarrow optimizer(args.optimizer\_config_t)$
      Compute loss: $L(\Theta)$
      Compute gradient: $\nabla(\Theta)$
      Update model: $\Theta \leftarrow optimizer_t.step(\nabla(\Theta), \Theta)$
    end
  end
end

Algorithm 1: Training multi-task model

Paraphrase Detection task uses Binary Cross-Entropy Loss as the loss function. With y as label (0 or 1) and p as model predicted probability. Its loss can be defined as:

$$L(\Theta) = -(y \log(p) + (1 - y) \log(1 - p)) \tag{2}$$

Semantic Textual Similarity task uses Mean Squared Error(MSE) as loss function. y_i is the label score and $\hat{y_i}$ is the model predicated score. For a batch of D data points. Their loss can be defined as:

$$L(\Theta) = \sum_{i=1}^{D} (y_i - \hat{y_i})^2 \tag{3}$$

5 Experiments

5.1 Data

In this project we uses the dataset provided in default project codebase Stanford-CS-224N (2023). Which includes:

SST-5 Stanford Sentiment Treebank consists of 11,855 single sentences from movie reviews extracted from movie reviews. Each phrase has a label of negative, somewhat negative, neutral, somewhat positive, or positive. Data split as 8544 train, 1101 dev, 2210 test.

QQP A subset of Quora Dataset consists of question pairs with labels indicating whether particular instances are paraphrases of one another. Data split as 141,506 train, 20,215 dev, 40,431 test.

STS-B SemEval STS Benchmark Dataset consists of 8,628 different sentence pairs of varying similarity on a scale from 0 (unrelated) to 5 (equivalent meaning). Data split as 6,041 train, 864 dev, 1,726 test.

5.2 Evaluation method

Sentiment Analysis: Measures model prediction accuracy against labels (0-5) in SST-5 dataset.

$$sst\_accuracy = \frac{sum(correctly\_predicted)}{total\_number\_of\_data} \tag{4}$$

Table 1: Model Parameters Comparison

Model	sst_lr	para_lr	sts_lr	Batch Size	Para Batch Size	Dropout
SLT	1e-6	1e-5	1e-5	32	32	0.3
MTL + CLB + SepInput	2e-6	2e-6	2e-6	32	32	0.3
MTL + CLB + ConcatInput	2e-6	2e-6	2e-6	30	150	0.5
MTL + SLB + ConcatInput	1e-6	3e-6	2e-6	30	150	0.5
MTL + SLB + ConcatInput + SelfAttn	1e-7	1e-5	1e-6	30	150	0.5

Paraphrase Detection: Measures model prediction accuracy against labels (0-1) in QQP dataset.

$$para\_accuracy = \frac{sum(correctly\_predicted)}{total\_number\_of\_data} \tag{5}$$

Semantic Textual Similarity: Measures Pearson correlation coefficient between model predicted score (x_i) and label scores (y_i) in STS-B dataset.

$$correlation = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{6}$$

5.3 Experimental details

The following experiment groups are run to compare different model architecture and training algorithms. This is not an exhaustive list of the experiments we have tried. For example, we have also experimented cosine similarity approach and applying gradient surgerys. But the following groups are selected because each of these experiments is a milestone with major discovery that leads to the final model. Table 1 shows the hyperparameters that are used when training these models. sst_lr, para_lr and sts_lr are the each individual learning rate for SST, Paraphrase Detection and STS.

- **SLT**: Single-task Learning
  - 3 Pre-trained BERT Models with a linear layer. Each model is fine-tuned on the individual sub-task.
- **MTL + CLB + SepInput**: Multi-task Learning with Combined Loss Backprop
  - MTL: Multi-task model using shared BERT encoder and a liner layer per task for prediction
  - CLB: During training run back-propagate on the combined losses of 3 sub-tasks.
  - SepInput: For tasks that have a pair of setences as input, encode them using BERT separately and then concatenate two output embeddings to feed into linear layer.
- **MTL + CLB + ConcatInput**: Multi-task Learning with Combined Loss Backprop and Input Concatenation
  - On top of MTL+ CLB, concatenate the sentence pairs into a single sentence and feed it to BERT encoder.
- **MTL + SLB + ConcatInput**: Multi-task Learning with Separate Loss Backprop and Input Concatenation
  - On top of MTL + CLB + ConcatInput, instead of CLB we run separate loss backpropogation on each sub-task, also using different optimizers to allow different learning rate and regularization configs.
- **MTL + SLB + ConcatInput + SelfAttn**: Multi-task Learning with Separate Loss Backprop, Input Concatenation with a self-attention in task-specific decoder.
  - On top of MTL + SLB + ConcatInput, added a self-attention layer in each task-specific decoder.

5.4 Results

5.4.1 Test Set Result

Our best model (MTL + SLB + ConcatInput + SelfAttn) achieved the following scores on test set.

- SST test Accuracy: **0.493**
- Paraphrase test Accuracy: **0.867**
- STS test Correlation: **0.867**
- Overall test score: **0.742**

5.4.2 Dev Set Results

Table 2 has shown the performance comparison for each model mentioned in 5.3. We noticed changing SepInput from ConcatInput has shown significant improvement on Paraphrase Accuracy and STS Correlation. By changing from CLB to SLB the slower converged task (Paraphrase Detection) improves. And finally by adding self attention layer in decoder each task is able to learn its own optimized pooling weights thus improves the overall quality on each task. More analysis and comparison is discussed in section 6.

Table 2: Model Performance Comparison on Dev Dataset

Model	SST Acc	Paraphrase Acc	STS Corr	Average
SLT	0.524	0.780	0.322	0.542
MTL + CLB + SepInput	0.515	0.761	0.372	0.549
MTL + CLB + ConcatInput	0.499	0.805	0.849	0.717
MTL + SLB + ConcatInput	0.487	0.828	0.847	0.720
MTL + SLB + ConcatInput + SelfAttn	0.504	0.869	0.868	**0.747**

6 Analysis

Multi-Task vs Single-Task We noticed task with sparse dataset for example STS benefits more from the Multi-task learning, due to other similar tasks' influence on the shared encoder.

SepInput vs ConcatInput For tasks that uses sentence pair as input, concatenating the pair of the sentence as single sentence input to BERT has a big leap in quality. It's likely due to BERT pretraining tasks e.g. "next sentence prediction" handles sentence pairs inputs as $CLS + sentence1 + SEP + sentence2 + SEP$. So the pretrained BERT understand this format better.

Combine Loss BackProp vs Separate Loss Backprop Separate loss backprop allows different learning rates and regularization, allowing different decoders to converge simultaneously. This capability improves the paraphrase detection performance in this MTL model.

With vs W/O Self Attention Adding self-attention in each task-specific decoder has a general quality lift on every task, it makes the model to automatically learn the best pooling method for each specific task thus increases decoders' learning capacity.

7 Conclusion

We developed a BERT-based multi-task learning model with task-specific decoders, fine-tuned on three tasks: Sentiment Analysis, Paraphrase Detection, and Semantic Textual Similarity. Feeding concatenated sentence to BERT encoder shown big quality lift on QQP and STS. Having separate loss backpropogation and optimizer on each sub-task has helped on solving the data imbalance and different coverge issue. We have also enhance the decoder's capacity by adding self-attention layer to each decoder, it has shown performance improvement across all tasks. The final model combined these 3 strategies results in the best overall performance.

References

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

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.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 350–356, Geneva, Switzerland. COLING.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732, Beijing, China. Association for Computational Linguistics.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. *CoRR*, abs/1901.11504.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, pages 79–86. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084.

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.

Anders Søgaard and Yoav Goldberg. 2016. Deep multi-task learning with low level tasks supervised at lower layers. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 231–235, Berlin, Germany. Association for Computational Linguistics.

Stanford-CS-224N. 2023. Default final project: minbert and downstream tasks. Https://web.stanford.edu/class/cs224n/project/default-final-project-bert-handout.pdf.

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.