Beyond BERT: Deepening Natural Language Understanding with Multi-Task Learning and Advanced Embedding Techniques

Stanford CS224N Default Project
Mentor: Hans Hanley. No external collaborators. Not sharing project

Varun Kutirakulam
Department of Computer Science
Stanford University
vkutira@stanford.edu

Mohammed Majid
Department of Computer Science
Stanford University
mmajid98@stanford.edu

Matt Peng
Department of Computer Science
Stanford University
mapeng3@stanford.edu

Abstract

In this paper, we 1) implemented a minimal version of BERT (minBert) and evaluated its performance on the sentiment analysis task and 2) fine-tuned minBert to simultaneously perform better on sentiment analysis, paraphrase detection and semantic textual similarity tasks. We combined the techniques of language model pre-training and multi-task learning with strong influence from the Multi-Task Deep Neural Network (MT-DNN) architecture Liu et al. (2019), and incorporated novel changes to task-specific training processes. This included modifying the task-specific classification layers, using cosine similarity to better perform paraphrase detection, and enhancing output embeddings of minBert to improve the performance of the semantic textual similarity task. Our empirical results indicate significant improvements beyond the baselines that we compared against, proving that our approach holds potential key analysis for future works.

1 Introduction

Learning robust word embeddings is a critical part of building natural language processing (NLP) systems that perform well on multiple NLP tasks. Two popular approaches to do this include language model pre-training and multi-task learning.

Language model pre-training has been extremely effective in improving model performance on many NLP tasks like natural language inference Bowman et al. (2015) and paraphrasing Dolan and Brockett (2005). By pretraining on a large corpus of text data, models learn general-purpose representations of linguistic features and patterns such as syntax, semantics, and pragmatics Raffel et al. (2020). One of the most influential pre-trained language models is BERT Devlin et al. (2019).

Multi-Task (MT) learning, on the other hand, focuses on the idea that models trained for specific NLP tasks may learn shared representations Crawshaw (2020). Hence, in applying MT learning to language models, a model can be trained on all NLP tasks simultaneously rather than the traditional approach of fine-tuning the model on each NLP task individually. This allows us to efficiently utilize the available data while leveraging the regularization effect to alleviate over-fitting to a specific task, thereby learning shared representations that are universal across multiple NLP tasks.

In this work, we implemented a baseline minimal BERT (minBert) model and modified it to adopt a Multi-Task learning architecture and training procedure called Multi-Task Deep Neural Networks (MT-DNN) Liu et al. (2019). This improved its performance on the three target tasks: sentiment analysis, paraphrase

detection, and semantic textual similarity. Note that we consider the modified minBert as our second baseline (MT-DNN (Base)). Additionally, we investigated novel changes that can enhance the model, such as adding Pre-Layer Normalization (pre-norm) to self-attention, enhancing contextual embeddings to extract task-specific representations, adding task-specific feedforward layers, and integrating cosine similarity to improve paraphrase detection Reimers and Gurevych (2019a). Multiple optimizers were also explored to improve the training procedure. Our empirical results as shown in Table 1 indicate that among all approaches we tried, combining MT learning with additional task-specific feedforward layers significantly improved the performance for all tasks. Also, enhancing contextual embeddings of sentence pairs by concatenating the difference and Hadamard product of the output embeddings before passing through the task-specific layers significantly improved our model performance on the semantic textual similarity task. Finally, using the cosine similarity loss as the objective for paraphrase detection significantly improved the model's performance.

	Tasks		
Model	Sentiment Analysis (Accuracy)	Paraphrase Detection (Accuracy)	Semantic Textual Analysis (Pearson)
minBert	0.375	0.266	0.599
MT-DNN (Base)	0.408	0.751	0.308
Final model	0.503	0.835	0.846

Table 1: Performance scores for the minBERT, MT-DNN (Base) implementation, and our improved Final model on the three tasks with dev datasets. Our Final model performs uniformly better than the two baselines.

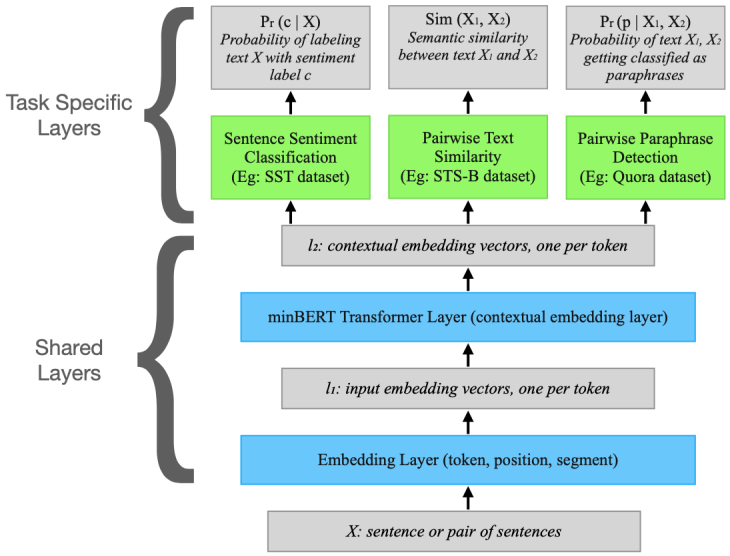

Figure 1: Overview of the MT-DNN (Base) model

2 Related Work

Multi-Task Learning:

Multi-task learning typically concerns training a single model to perform well on a variety of tasks. While single-task training of BERT involves fine-tuning the model on a per-task basis, a common technique in multi-task learning is to add the losses for each of the unique tasks in the current batch of training data Bi et al. (2022). For example, calculating the loss for gradient updates, assuming n total tasks, could look like:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{task\,1} + \mathcal{L}_{task\,2} + ... + \mathcal{L}_{task\,n} \quad (1)$$

An important thing to note is that, depending on the multi-task learning process, gradient directions could be vastly different and affect the overall training process. Thus it is important to consider methods to avoid conflicts of gradients among the training tasks to ensure a smoother learning process.

Multi-Task Deep Neural Network (MT-DNN):

MT-DNN Liu et al. (2019) shown in Figure 1 has the strongest influence on our work as our training process was built on top of the MT-DNN training algorithm. The MT-DNN algorithm from the original paper combines the training of four NLP tasks: Single-Sentence Classification, Text Similarity, Pairwise Text Classification, and Relevance Ranking. The architecture of the MT-DNN is composed of two parts. The lower layers are shared and learn generic text encodings that are fine-tuned to support all NLP tasks. Task-specific layers are added on top that generates task-specific representations, followed by the operations necessary for classification, similarity scoring, or relevance ranking. During training, the task-specific datasets are shuffled, and the model is updated according to the task-specific objective. This approximately optimizes the sum of all multi-task objectives. One limitation of the work was that it did not provide significant attention to improving the embeddings to amplify task-specific features that were returned from the shared layers of the MT-DNN model before passing them through the classification layers. While we followed the training process of MT-DNN, we improved upon the model architecture and integrated other novel NLP strategies to address the limitation noted and build a more performant extension of this work.

3 Approach

We modified the minBert that follows the MT-DNN model and further updated its architecture by adding novel changes to improve its performance for the three NLP tasks. As shown in Figure 2, our model consists of four key components. Starting from the bottom, our model contains 1) our baseline MT-DNN implementation built on minBert with post-layer normalization, 2) the enhancement layer that updates contextual embeddings to extract task-specific semantic knowledge, 3) the feedforward network layer that learns task-specific representations to fine-tune the performance of the associated task, and 4) the task-specific classification layers that predict the scores for given task inputs. In addition to the model architecture, we identified optimizers and task-specific objective functions that efficiently train the model to best perform the three NLP tasks.

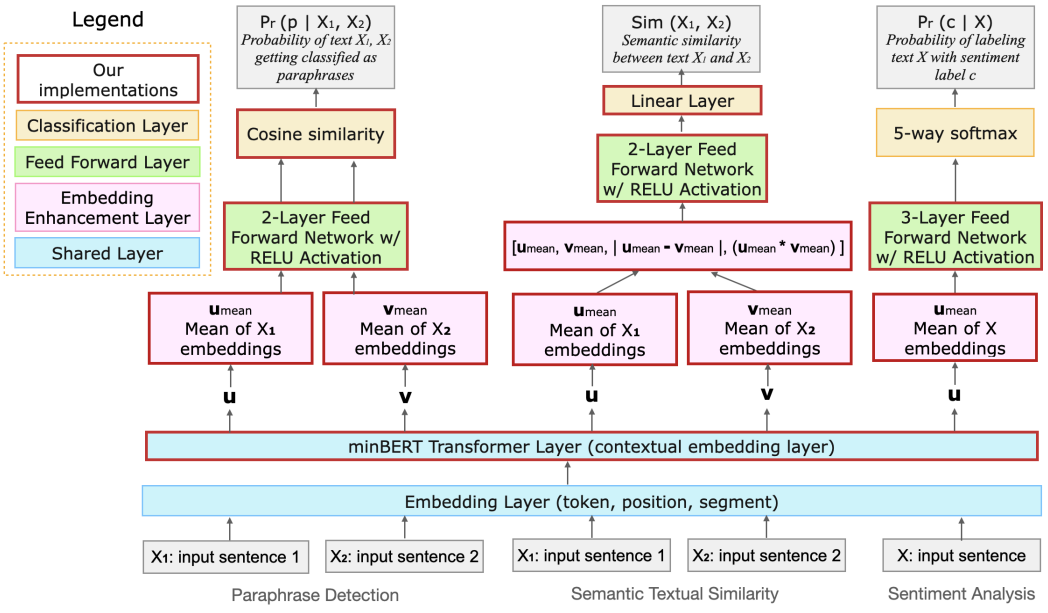

Figure 2: Overview of our fully improved model. The components highlighted in red indicate the novel differences between our model and MT-DNN.

3.1 Baseline MT-DNN Model

For our baseline MT-DNN model, we followed the model architecture presented in Liu et al. (2019) and used our minBert implementation to capture the shared representations. In addition, we implemented individual classification layers to predict task-specific scores for given inputs. A key distinction between our implementation and the original model is that the original model is based on BERT while ours is based on our minBert implementation.

3.1.1 Pre-Layer Versus Post-Layer Normalization

Our first design decision for our model was whether to use pre or post-layer normalization for our minBERT model. This choice determines where the layer normalization is applied with respect to the multi-head attention and feedforward layers. Many works have noted stronger performance with pre-norm, finding that it is more effective in capturing long-range dependencies and modeling complex linguistic structures (Xiong et al. (2020), Gururangan et al. (2020)), which is why we explored the approach by implementing a pre-norm transformer layer in our minBert implementation.

3.2 Embedding Enhancement

The shared layer (minBert) outputs contextual embeddings corresponding to the input sentence. Rather than the traditional approach of using just the CLS token embeddings to perform tasks, we decided to take the mean of all token embeddings to capture additional semantic knowledge of the input. In addition, for sentence pairs, we create concatenated embeddings containing the original embeddings, the differences, and the Hadamard product of the embeddings to feed into the task-specific feedforward layers. Consider sentence embeddings u and v. We take the element-wise absolute difference $|u - v|$, the element-wise product $u * v$, and the original embeddings to create an embedding that contains u, v, $|u - v|$, and $u * v$.

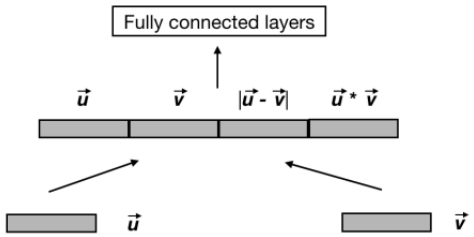

Figure 3: Process of creating our concatenated embeddings.

The difference embedding, $|u - v|$, can be useful in capturing dissimilarities in word order or negation. On the other hand, element-wise product embedding, $u * v$, can capture the similarity in meaning between the two sentences. The combination of all the above embedding enhancements has previously been shown to improve its ability to detect paraphrases and measure semantic similarity (Cer et al. (2018), Reimers and Gurevych (2019a), Lan and Xu (2018)).

3.3 Task-Specific Feedforward Network

The original MT-DNN architecture uses a single linear transformation layer as a classifier for each NLP task. We introduced a task-specific feedforward network with non-linear activation between the layers. By doing this, our model is able to capture task-specific information, while still benefiting from the shared representations learned by the embeddings. Empirically, we noticed a huge improvement in our performance with this change and also note that prior works have also seen this result with this change (Liu et al. (2017), Long and Wang (2015)).

3.4 Cosine-Similarity Loss

A technique we used to specifically improve the performance of the paraphrase detection task was to implement and employ cosine-similarity loss Reimers and Gurevych (2019b). As opposed to concatenating the output embeddings, we push the original embeddings for the sentence pairs u, and v through the feedforward task-specific layers. Next, the cosine similarity score of the outputs is calculated, which acts as the objective score that is maximized if the inputs are paraphrases and minimizes the score otherwise.

3.5 Optimizer

In addition to the model architecture, we also investigated optimizer options such as AdamW, Yogi, and Stochastic Gradient Descent (SGD) Loshchilov and Hutter (2019). The AdamW optimizer Kingma and Ba (2017) is a strong candidate as it is an extension of SGD that uses adaptive learning rates for each parameter and momentum to accelerate convergence. Additionally, it also supports an additional weight decay term to help prevent overfitting. However, one of its drawbacks, when compared to SGD, is that AdamW can be more sensitive to outliers and not generalize the model as well as SGD. We also experimented with Yogi optimizer Zaheer et al. (2018) as it was shown to accelerate convergence. It does so by being an additive adaptive method, as opposed to multiplicative, that uses momentum correction to help handle noisy gradients. Yogi is able to change the learning rate in a more controlled fashion compared to AdamW due to the fact that the velocity term is largely based on the square of the gradient as described in the original paper Zaheer et al. (2018).

3.6 Training Procedure

With our model fully defined, we now discuss the training procedure for our model architecture. We first load pre-trained BERT weights into the minBert model. Next, we follow the MT-DNN training procedure and pack task-specific datasets into mini-batches. These batches are then grouped and shuffled. In each epoch, a mini-batch b_t of data is randomly selected and the model is updated by the task-specific objective function. This approximately optimizes the sum of all the multi-task objectives following the common MT learning framework. Pseudo-code for our training process can be found in Algorithm 1.

Algorithm 1 MT-DNN Training Process

Initialize model parameters θ randomly.
Load pre-training weights for the shared layers (i.e., the token embedding and minBERT layer).
Set the max number of epoch: max_{epoch}
Prepare the data for T tasks.
for t in $1, 2, ..., T$ **do** Pack the dataset t into mini-batch: D_t
Merge all the datasets: $D = D_1 \cup D_2 ... \cup D_T$
Shuffle the mini-batches in D
for b_t in D **do**
    Compute loss: $L(\theta)$
        $L(\theta)$ = Eq. 2 //$for\ sentiment\ analysis$
        $L(\theta)$ = Eq. 3 //$for\ paraphrase\ detection\ and\ semantic\ textual\ similarity$
    Compute Gradient: $\nabla\theta$
    Update model: $\theta = \theta - \nabla\theta$
Finish

3.6.1 Task Specific Loss Functions

We selected our task-specific loss functions by choosing objectives that correspond directly with the NLP task at hand. For sentiment analysis, we used cross-entropy loss,

$$L_{CE} = -\sum_{c} \mathbb{I}(X, c) \log(P_r(c|X)) \tag{2}$$

where $\mathbb{I}(X, c)$ is the binary indicator if class label c is the correct classification for X and $Pr(.)$ is the output received from the classification layer.

For both paraphrase detection and semantic textual analysis, we used mean squared loss (MSE),

$$L_{MSE} = (y - Sim(X_1, X_2))^2 \tag{3}$$

where y is the annotated real-valued score for each sentence pair (X_1, X_2) and $Sim(.)$ is the output similarity score from the classification layer.

4 Experiments

4.1 Data

We used the Stanford Sentiment Treebank (SST) Socher et al. (2013) and the CFIMDB dataset for the sentiment analysis task. For the SST dataset, we trained on 8,544 phrases and evaluated 1,101 phrases from the dev dataset. Each phrase has a label of negative, somewhat negative, neutral, somewhat positive, or positive. From the CFIMDB dataset, we trained on 1,701 examples and evaluated 245 examples from the dev dataset. We used the Quora dataset[1] for the paraphrase detection task and trained on 141,506 examples and evaluated 20,215 examples from the dev dataset. Each paraphrase pair is either classified as "Yes" or "No". Lastly, we use SemEval STS dataset[2] for the semantic textual analysis task and trained on 6,041 examples and evaluated 864 examples from the dev dataset. Each paraphrase pair has a degree of similarity score on a scale from 5 (same meaning) to 0 (not at all related).

4.2 Evaluation Method

The evaluation method used for paraphrase detection and sentiment analysis was the accuracy score, while the Pearson correlation coefficient was used to evaluate semantic textual similarity.

4.3 Experimental Details

The minBert model was first implemented and trained to perform three target tasks: sentiment analysis, paraphrase detection, and semantic textual similarity. The model performance was evaluated using the task-specific dev datasets mentioned above, and the results have been shared in Table 1. The minBert model was modified to replicate the MT-DNN model, which acted as our second baseline. Performance of our MT-DNN model was also evaluated for the three tasks using the same dev datasets, and results have been shared in Table 1 as well.

Next, we ran ablation studies and experimented by adding the different components mentioned in our Approach section on top of the MT-DNN baseline one at a time to determine the magnitude of improvement in performance for each one of our components. We ran the experiments in Table 2 in sequential order starting from the top. The experiments were designed such that the lower shared layers were first experimented upon to finalize the backbone, followed by the task-specific layers that were modified to improve task performance. Each row in Table 2 specifies an ablation exercise that was performed, where the first model within each row is used as a baseline (Base). Following are the ablation exercises that were carried out, with their results shared in Table 2.

- MT-DNN w/ Post-Layer Normalization (Base) **vs** w/ Pre-Layer Normalization
- Task-Specific Single-Layer feedforward Network (Base) **vs** 2-Layer feedforward Network
- Perform tasks based on CLS token embeddings (Base) **vs** Mean of all contextual embeddings
- No embedding Enhancements (Base) **vs**

  Concatenated Embeddings (Concat) for both sentence pair tasks **vs**

  Cosine Similarity + MSE Loss (Cosine) for both sentence pair tasks **vs**

  Concat for Sem. Text Similarity task & Cosine for Paraphrase Detection task
- AdamW (Base) **vs** Yogi **vs** SGD optimizers

Once the model was finalized, it was trained on the training datasets mentioned above and evaluated to determine its performance against the two baselines (minBert and MT-DNN (Base)). The results of the experiment are shared in Table 1.

The hyper-parameters were the same across all experiments to maintain the consistency of the results. For each experiment, the models were pre-trained for 10 epochs with a learning rate of $1e^{-3}$ and a dropout probability of 0.3. The fine-tuning procedure was run for 10 epochs with a dropout probability of 0.3 as well, but with a learning rate of $1e^{-5}$. The Final model was trained on an additional 10 epochs to achieve the results shared in Table 1 and Table 3. The random seed was set to 11711 for all experiments.

[1] `https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs`

[2] `https://aclanthology.org/S131004.pdf`

4.4 Results

The results of our ablation experiments are provided in Table 2, which confirmed that following the MT-DNN architecture significantly improved model performance when compared to minBert. In addition, task-specific embedding enhancements and classification layers predictably provided the best performance gains on top of the MT-DNN baseline. However, it was surprising that contrary to the literature review, pre-layer normalization performed significantly worse than the baseline. Lastly, we observed that both AdamW and Yogi optimizer Zaheer et al. (2018) had very similar performance.

	Tasks			
Model	Sent. Analysis	Para. Detect.	Sem. Text Analysis	% Improv
minBERT (Base)	0.375	0.266	0.599	0
MT-DNN + Post-norm	0.408	0.751	0.308	**32.2**
MT-DNN + Post-norm (Base)	0.408	0.751	0.308	**0**
MT-DNN + Pre-norm	0.352	0.744	0.204	-22.6
1-Layer FeedForward (Base)	0.408	0.751	0.308	0
2-Layer FeedForward	0.414	0.743	0.329	**2.7**
CLS-Token Embed (Base)	0.414	0.743	0.329	0
Mean Embed	0.452	0.774	0.369	**8.5**
No Enhancement (Base)	0.452	0.774	0.369	0
Concat (Para + Sem)	0.445	0.445	**0.835**	16.9
Cosine (Para + Sem)	0.408	**0.809**	0.642	22.6
Cosine (Para) + Concat (Sem)	0.466	0.798	0.812	**42.1**
AdamW (Base)	**0.466**	0.798	0.812	**0**
Yogi	0.425	**0.809**	0.827	-1.9
SGD	0.441	0.738	**0.829**	-3.6

Table 2: Results for ablation experiments. Each experiment quantitatively validates the model architecture for optimal performance.

The Final model was evaluated on dev datasets along with minBert and MT-DNN (Base). The performance scores have been reported in the Introduction section in Table 1. The performance of the Final model on the test datasets for the three target tasks can be found in Table 3 below.

	Tasks		
Model	Sentiment Analysis (Accuracy)	Paraphrase Detection (Accuracy)	Semantic Textual Analysis (Pearson)
Final model	**0.532**	**0.818**	**0.841**

Table 3: Performance scores for the Final model on test datasets

5 Analysis

In this section, we offer further analysis of embedding enhancements and share an ablation analysis of the different optimizers we used for our model.

5.1 Enhanced Embeddings: A Case Study

During ablation experiments, we noticed that the semantic textual similarity task gained peak performance when the output contextual embeddings of the sentence pairs were enhanced by concatenating the difference and element-wise product of the embeddings to form a new embedding that was passed through the feedforward network prior to classification. Concatenating the difference of the embeddings allows the feedforward network to capture and maximize the dissimilarities between the sentence pairs, while the element-wise product captures the similarities between the sentence pairs. These new features can be amplified by the feedforward network during training to learn their representations. Following is a sentence pair that was a part of our dataset used to evaluate the semantic textual similarity task.

Sentence 1: A man dives off of a cliff.

Sentence 2: A man is riding a motorcycle.

True Similarity score: **0.66**

MT-DNN (Base) - Predicted Similarity Score: **2.61**

Final Model - Predicted Similarity Score: **0.67**

We can see that the baseline MT-DNN model incorrectly predicted the sentence pair to be somewhat similar, most likely because the subject of the sentences ("man") and their semantic structures are very similar. The sentence length is small and equal as well, which is another primitive feature that is most likely captured by the MT-DNN base model. However, since our model enhances the embedding pairs to amplify critical differences, it pays more attention to the action verb of the sentence pairs, which are dissimilar and affect different objects in the sentences. Hence, our model can correctly predict that the sentence pair are actually dissimilar. Quantitative analysis validates our understanding as well since the ablation studies revealed that the Pearson score of semantic textual similarity task increased by 50% during evaluation when the embeddings were enhanced, as shown in Table 2.

5.2 Ablation: Optimizer Choice

We noticed a difference in the convergence of our experimental results when changing the optimizer to train our model. Different optimizers have their strengths and weaknesses, which can significantly impact the performance of the model. Additionally, since the datasets for each of our three tasks were unbalanced, we noticed fluctuations in performance when using different optimizers. Figure 4 shows how different optimizers performed over 10 epochs during training. We were especially interested to study the difference in performance between the Yogi Zaheer et al. (2018) optimizer and the AdamW optimizer since Yogi has been proposed to improve convergence speed and stability over AdamW. While the graph displays the consistency in Yogi's results, both achieved similar performance during training. This led us to side with the AdamW optimizer since a large corpus of previous works validate its success. It is possible that Yogi would've performed better with additional hyper parameter tuning and a more balanced data distribution. SGD took significantly longer to converge and was not able to match the performance of the other two optimizers, so we chose not to pursue it any further.

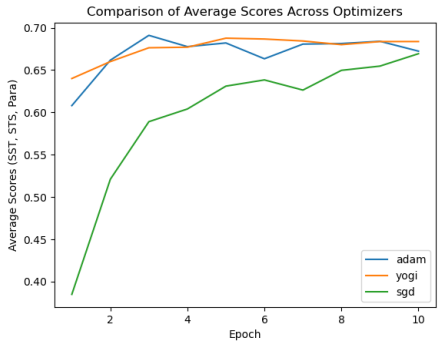

Figure 4: Results of our optimizer ablation experiment.

6 Conclusion

We have built a multi-task learning model that extends the MT-DNN architecture and performs well on sentiment analysis, paraphrase detection, and semantic textual similarity NLP tasks. Additional components of our model beyond the MT-DNN include an embedding enhancement layer that modifies output embeddings to amplify task-specific features, task-specific feedforward networks that can learn additional representation, and updated classification layers that better predict sentence pair similarities. Our experiments show that the model uniformly beats our original minBERT model and our baseline MT-DNN model. We noticed that the performance of our model on certain tasks was also influenced by the amount

of data that our model was trained upon. Future works could investigate the link between disproportional task-specific dataset size and its effects on multi-tasking NLP systems. We also believe that our model encourages further research and investigation into techniques that enhance contextual output embeddings to amplify task-specific features and improve the performance of other NLP tasks.

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*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. *CoRR*, abs/1803.11175.

Michael Crawshaw. 2020. Multi-task learning with deep neural networks: A survey.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Wuwei Lan and Wei Xu. 2018. Neural network models for paraphrase identification, semantic textual similarity, natural language inference, and question answering. *CoRR*, abs/1806.04330.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial multi-task learning for text classification.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding.

Mingsheng Long and Jianmin Wang. 2015. Learning multiple tasks with deep relationship networks. *CoRR*, abs/1506.02117.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer.

Nils Reimers and Iryna Gurevych. 2019a. Sentence-bert: Sentence embeddings using siamese bert-networks.

Nils Reimers and Iryna Gurevych. 2019b. Sentence-bert: Sentence embeddings using siamese bert-networks. *ArXiv*, 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.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. 2020. On layer normalization in the transformer architecture.

Manzil Zaheer, Sashank J. Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. 2018. Adaptive methods for nonconvex optimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 9815–9825, Red Hook, NY, USA. Curran Associates Inc.