Finetuning minBERT for Downstream Tasks

Stanford CS224N Default Project

Pete Rushton
Department of Computer Science
Stanford University
`prushton@stanford.edu`

Tyler Nichols
Department of Computer Science
Stanford University
`catpics@stanford.edu`

Abstract

Models that can infer semantic understanding from text are applicable to problems spanning many domains. We investigate the application of several techniques to a pretrained BERT model to maximize multitask accuracy when performing sentiment analysis, paraphrase detection, and similarity evaluation. We present implementations of two extensions to the BERT model and apply them to these three tasks. Of these extensions, a novel implementation of question-answering yielded outstanding results.

1 Key Information to include

Mentor: Anuj Nagpal. External collaborators: none. Sharing project: not applicable.

2 Introduction

Building upon the original BERT model presented by Devlin et al [1], we experiment with two extensions based upon a question-answer format for inputs originally proposed by McCann et al [2] and a triplet loss function proposed by Henderson et al [3]. Although both approaches yield improvements over baseline across all three tasks, we found that the improvement wrought by reformulating inputs as question-answer prompts was dramatic.

In this paper, we detail our approaches to implementing both of these concepts, our experiments, the results of our experiments, and our interpretation of the results.

3 Related Work

While the many important contributions to the field of multi-task NLP models are too numerous to cover here, we highlight three works that have been particularly influential on our experimentation.

Devlin et al (2018) [1] created the original BERT model, which achieved state-of-the-art performance on 11 NLP tasks compared to existing architectures, and it is the foundational pre-trained model upon which our experimentation has been based. The authors demonstrated that the efficacy of the pre-trained model obviated the need for more complicated and/or task-specific architectures. Furthermore, we harnessed BERT's known strong performance in question-answer tasks after fine-tuning in order to create our most performant model, Q&ABERT.

The second is McCann et al (2018) [2]. Their approach casts "all tasks as question answering over a context... without any task-specific modules or parameters," with the aim of equalling benchmark performance across 10 separate NLP tasks. As discussed below, their reframing of separate tasks into a single question-answering multitask framework was highly influential in creating Q&ABERT.

Lastly, Henderson et al [3] introduced the multiple negatives loss function, which inventivizes embeddings that are in close proximity in embedding space when the inputs to the model are

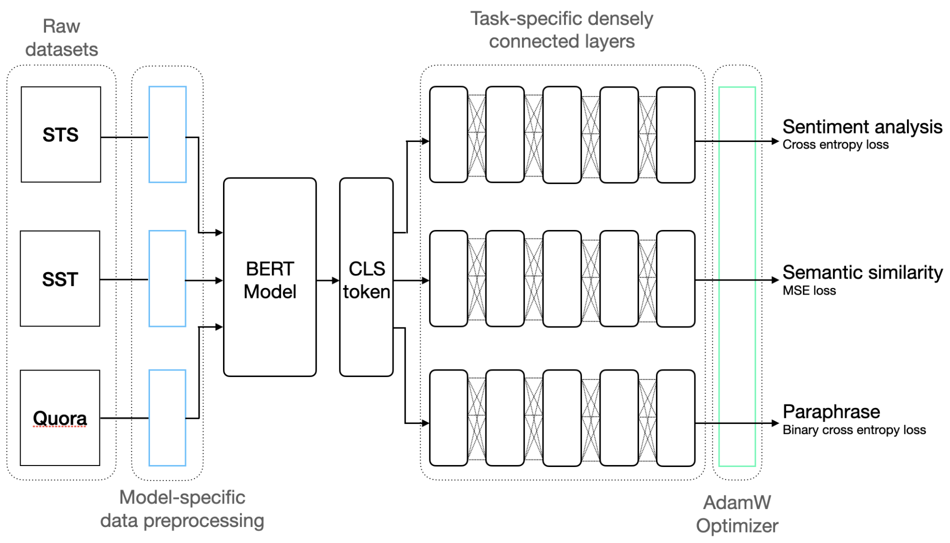


Figure 1: Simplified diagram of model architecture

similar while conversely incentivizing embeddings to be far apart from each other when the inputs are dissimilar. This was shown to substantially improve the quality of learned embeddings. We implemented this loss function and applied it to our 3 tasks, as described below.

4 Approach

4.1 Model Architecture

Our experimental efforts focused upon enhancements to BERT via finetuning and placed less emphasis on the task-specific output heads. As such, our architecture remained relatively simple and consistent across our experiments.

Our models utilize a single BERT model shared by three prediction heads, each one trained for a single task, as illustrated in Figure 1. Each of the three task-specific prediction heads consist of 5 densely connected feed-forward neural networks. Each FFN consists of 4 layers wherein output from a linear layer is passed through a layer norm, dropout is applied, and then ReLU is applied, the output of which is used as input to the next layer. Each prediction head has a final linear layer that projects the high-dimensional output from the prior 4 layers down to a set of logits of the requisite dimensions. Tokenized sentence inputs are converted to CLS tokens by the BERT model uniformly across all three tasks and the CLS tokens are used as input to each of the three aforementioned prediction heads.

On each epoch, we train over the data sets for each of the three tasks in sequence, beginning with the SST data set, followed by the STS data set, and concluding with the Quora data set. In doing so, we sample in order to overcome biases within the data sets, as further detailed in section 4.2.3. With the exception of our baseline model, we also perform some model-specific preprocessing as described in section 4.2. We also used only a subset of the Quora data set at each epoch as described in 5.3

We use cross entropy as our loss function for the SST task, mean-squared error loss for the STS task, and binary cross entropy for the Quora paraphrase task. Our models use AdamW as the optimizer for all 3 tasks. After loss is computed for each task, we backpropagate over the corresponding task-specific FFN prediction head as well as the entire shared BERT model, according to the `finetune` option supplied in the DFP starter code.

4.2 Data Transformation

As part of our experimentation, we transformed the provided data sets in two key ways: first, by reformatting the data for use in a triplet loss function, and second, by reframing data from each of the three tasks in a question-answer format. We also discovered and corrected a bias in the distribution of labels within the training data that negatively impacted our models' ability to perform the tasks.

4.2.1 Triplet Loss Format

One experiment involved pretraining the shared BERT model using a custom triplet loss function.

To do so, we homogenized the SST, STS, and Quora data sets to a single, unified format, combined data from all three data sets, converted the data to embeddings using the task-agnostic shared BERT model, and structured the loss function to influence the distribution of embeddings produced by BERT.

The three data sets were homogenized into a unified data frame format consisting of:

- An anchor sentence
  - For SST and STS, every sentence is used as an anchor exactly once
  - For Quora, we limit the selection of anchor sentences to inputs from the original data set for which the associated prompt is positively labeled as being paraphrased
- A positive sample
  - For SST, this is a sentence with the same sentiment label
  - For STS, this is a sentence with a similar label in the range of either 0-2 or 3-5
  - For Quora, this is the paraphrase of the corresponding anchor
- Three negative samples from the same data set
  - For SST, these are sentences with different sentiment labels
  - For STS, these are sentences with labels from the opposite range of either 0-2 or 3-5
  - For Quora, we specifically include negative samples from the portion of the data from which we could not select anchor sentences
- One negative sample from each of the other two data sets relative to the anchor

We generated a set of these data frames spanning all available data in all three provided data sets, inputted them to the common BERT model, evaluated loss as defined in equation 1, and backpropagated over the BERT parameters.

As this loss function is designed to shape the distribution of CLS embeddings in embedding space, the loss is computed exclusively over the embeddings produced by the shared BERT model and only the BERT parameters were updated during backpropagation. We did not freeze the BERT parameters after training them with this loss function; in subsequent task-specific training loops, backpropagation was performed over the corresponding prediction head as well as the underlying shared BERT parameters.

$$\mathcal{L}_{minibatch}(\theta) = \sum_{i}^{A} \left[\max\left(0, d(\mathbf{x}_i, \mathbf{y}_i) - \log\left(\sum_{j}^{N} \exp\{d(\mathbf{x}_i, \mathbf{z}_j)\}\right)\right)\right] \tag{1}$$

In equation 1, A is the set of anchor sentences, $d(\cdot, \cdot)$ is an arbitrary vector distance function for which we chose to use Euclidean distance, and N is the set of negative samples for the corresponding anchor sentence. $\mathbf{x}_i$ represents the embedding of the current anchor sample, $\mathbf{y}_i$ represents the embedding of the corresponding positive sample, and $\mathbf{z}_i$ represents the embedding of a negative sample corresponding to the current anchor sentence.

4.2.2 Question-Context-Answer Format

Following the work of McCann et al (2018) [2], we reframed our 3 separate datasets and tasks into a single question-answering multitask framework, as shown in Table 1. This reframing is executed at the dataset preprocessing stage of our architecture, as illustrated in Figure 1. We refer to this model as Q&ABERT. The model pads and truncates each question-context-answer triple input as necessary to a vector of length 758. Thus a critical difference compared to our baseline is that, for the Quora and STS tasks, Q&ABERT receives a singular CLS token that embeds both sentences in a single

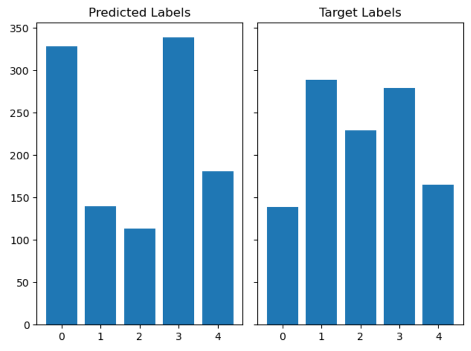
Quora or STS input, before passing them to the relevant task-specific FC layers. By contrast, our baseline for those 2 tasks receives 2 CLS tokens and contatenates them, before passing the result to the task-specific FC layers. We hypothesize that the question-context-answer triples will enable BERT to learn to produce CLS tokens for these tasks that better represent the answer-semantics of the task-question given the supplied context-input. Additionally, for all 3 tasks but particularly for Quora and STS, we hypothesize that the question-context-answer triple form will leverage BERT's abilities stemming from its Next Sentence Prediction training, which BERT's creators note was intended to "understand the relationship between two sentences"[1]. We note that the baseline model does not leverage BERT's pretraining in this way at all.

We highlight 2 significant architectural differences between Q&ABERT and McCann et al's model, which they called MQAN. First, the bulk of our architecture consists of a pretrained BERT model, whereas MQAN is founded on BiLSTMs. Second, Q&ABERT (like all our models presented in this paper) has output layers that are dedicated to each task, whereas MQAN had a single unified architecture for all 10 NLP tasks that they pursued.

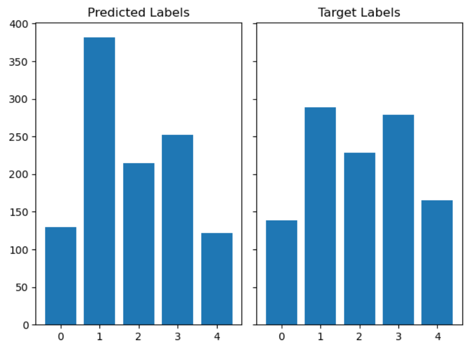
Table 1: Examples of re-framing our 3 separate tasks and datasets as a single question-answering multitask framework that converts each example into a (question, context, answer) triple

Question	**Context**	**Answer**
Is this review positive or negative?	Dramas like this make it human.	4
Does the first question paraphrase the second question?	First sentence: How can I master myself in geometry? Second sentence: How can I master geometry for the CAT-14?	No
Is the first sentence semantically equivalent to the second sentence?	First sentence: A kid is talking in class. Second sentence: A girl is going to class.	1.8

4.2.3 Biased Label Distribution



(a) Before correction



(b) After correction

Figure 2: Biased Distribution of Predicted Labels in SST Data

Our experiments revealed a substantial bias in the distribution of ground truth labels in the training data. See Table 3 for full details of the distributions. This bias appears to have disproportionately affected our baseline model performance on the dev data set of the SST 5-way sentiment classification task.

We addressed this bias by utilizing the `WeightedRandomSampler` class in PyTorch and by calculating weights for each label. Labels that occurred less frequently are weighted more heavily and labels that occurred more frequently are weighted more lightly. Weights are assigned per sample, and the weights sum to 1 for each label.

Figure 2a illustrates the result before making this correction, and shows a distribution of predicted labels that deviates from the ground truth labels substantially on the SST dev set. Figure 2b shows that re-weighting our training data to ensure even sampling across all labels during training allowed our model to produce a distribution of predicted labels during dev evaluation that much more closely approximates the distribution of ground truth labels.

4.3 Hyperparameter Search

We used manual hyperparameter searches to test various values for epoch count, dropout rate and learning rate. We also used the Weights&Biases MLOps platform[1] to perform some automated hyperparameter sweeps spanning our dropout and learning rates. While noting that time constraints necessitated a less-than-exhaustive approach, our preliminary conclusions are as follows: the optimal learning rate is $1e-5$, our architecture is somewhat sensitive to dropout rates, and dev set performance plateaus around 40 epochs. A visualization of our automated searches is provided in Figure 6.

5 Experiments

5.1 Data

As we are pursuing the Default Final Project, we use data sets provided by the staff. Our baseline results use the provided data as is. However, both of our primary methodological experiments transform the provided data as described in section 4.2.

5.2 Evaluation method

As required by the Default Final Project, we used the following evaluation metrics: accuracy for SST and Quora; Pearson correlation for STS.

In addition, we have attempted to make our work as comparable as possible to Devlin et al (2018) [1] by collecting other evaluation metrics in addition to those required for the DFP - namely, F1 for Quora and Spearman correlation for STS. That said, we note that we do not know the exact provenance of the DFP datasets and we cannot know the extent to which they may differ from those used by Devlin et al. We also made use of other evaluative tools to help guide our experimentation and assess the broader performance of our models. These tools included confusion matrices, prediction error visualizations for the STS regression task, and prediction distribution histograms for identifying potential biases. Where insightful, examples are included in this paper.

5.3 Experimental details

We used the full data sets for every epoch for SST and STS with sampling as discussed in section 4.2.3. We used only a sampled subset of the Quora data set at each epoch, because using its full size at each epoch would have resulted in slower training. We also hypothesize it may have overwhelmed the finetuning effects of other data sets.

Our hyperparameter search work led us to focus on 0.1 dropout rates and $1e-5$ learning rate, as well as favoring 50-100 epoch training cycles for our most promising models. These runs took approximately 1-2 minutes per epoch. We trained our models on AWS VMs and consumer machines. Very small differences in performance between the environments appear to result from these different hardware configurations.

5.4 Results

The results of our Triplet Loss and Q&ABERT experiments are summarized and compared to baseline and Devlin et al (2018) [1] results in table 2.

5.4.1 Baseline

Baseline model performance in the Quora task as measured by F1 was only slightly below prior results obtained from Devlin et al. We note that Devlin et al's SST task is not comparable with the DFP as the original BERT model was assessed on a binary sentiment analysis task using the SST dataset, whereas the DFP calls for a 5-way sentiment analysis. We observe that the baseline model performed very poorly compared to Devlin et al in relation to the STS task and as evaluated using Spearman correlation. Figure 3 shows the baseline model's performance in more detail.

[1] `www.wandb.ai`

Table 2: Metrics summary. *BERT results for SST represents accuracy on a binary classification, whereas Baseline, Triplet Loss and Q&ABERT represent 5-way classification. "-" indicates metric not applicable for task or not available. Q&ABERT Accuracy and Pearson metrics supplied from Gradescope results on the test dataset, whereas F1 and Spearman metrics are results from the dev dataset.

	SST	**STS**	**Quora**
Baseline			
Accuracy	0.440	-	0.765
F1	-	-	0.709
Pearson Corr. Coeff.	-	0.326	-
Spearman Corr. Coeff.	-	0.310	-
BERT (Devlin et al 2018)			
Accuracy	*0.949**	-	-
F1	-	-	0.721
Pearson Corr. Coeff.	-	-	-
Spearman Corr. Coeff.	-	0.865	-
Triplet Loss			
Accuracy	0.481	-	0.771
F1	-	-	0.727
Pearson Corr. Coeff.	-	0.453	-
Spearman Corr. Coeff.	-	0.429	-
Q&ABERT			
Accuracy (test dataset)	0.495	-	0.868
F1 (dev dataset)	-	-	0.836
Pearson Corr. Coeff. (test dataset)	-	0.843	-
Spearman Corr. Coeff. (dev dataset)	-	0.834	-

5.4.2 Triplet Loss

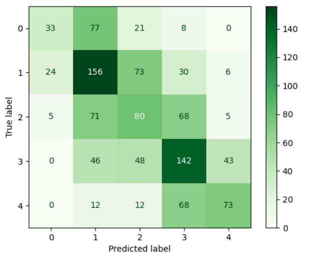
The addition of the triplet loss function to shape the distribution of embeddings produced by BERT yielded modest yet notable improvements across all three tasks. Table 2 details slight improvements on the SST and Quora tasks with a more notable boost on the STS task.

Compared to the confusion matrices and prediction error plot in figure 3, results from the triplet loss model show slightly substantially better clustering of predictions on the SST task in figure 4a, comparable clustering of prediction on the paraphrase task in figure 4b, and slightly better clustering of STS predictions in figure 4c.

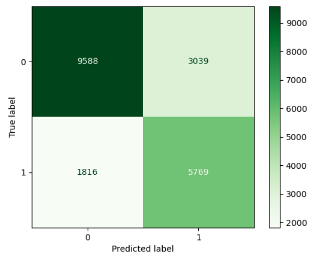
5.4.3 Q&ABERT

Reformulating the inputs as question-context-answer triples resulted in across-the-board improvements on all three tasks. The Q&ABERT model yields moderate improvements on SST and Quora and very significant improvements on STS. Our Q&ABERT test results were obtained on a model using the `finetune` option on the 3 datasets, with a learning rate of $1e-5$, a dropout percentage of 10% on the task-specific layers, and trained for 100 epochs.

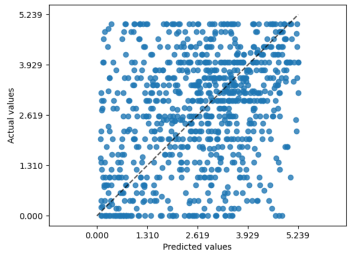
This model achieved SST accuracy of 0.495, STS Pearson correlation coefficient of 0.843, and Quora accuracy of 0.868 on the test data sets as determined by Gradescope. On dev data sets, it achieved a Spearman correlation cooefficient of 0.834 on STS and an F1 score of 0.836 on Quora. Figure 5 shows Q&ABERT's performance in more detail. The tighter clustering of predictions in Figure 5c is particularly noteworthy.



(a) SST Confusion Matrix

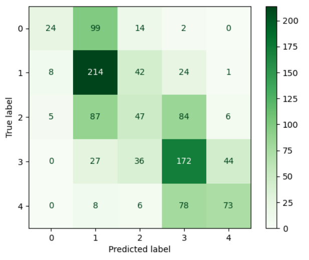


(b) Paraphrase Confusion Matrix

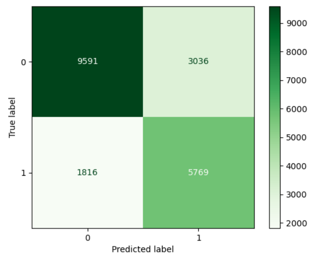


(c) STS Prediction Error

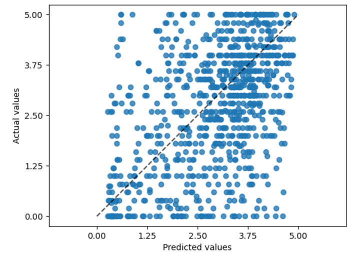
Figure 3: Baseline error analysis



(a) SST Confusion Matrix

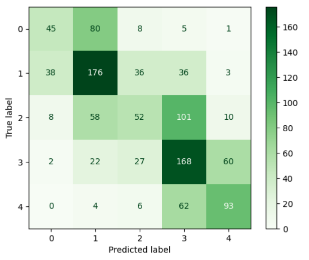


(b) Paraphrase Confusion Matrix

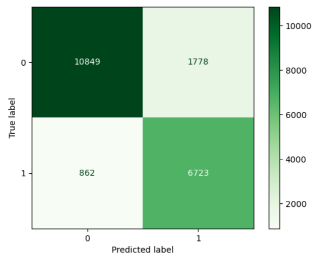


(c) STS Prediction Error

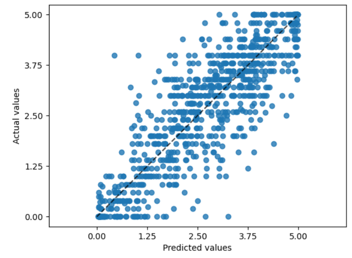
Figure 4: Triplet Loss error analysis



(a) SST Confusion Matrix



(b) Paraphrase Confusion Matrix



(c) STS Prediction Error

Figure 5: Q&ABERT error analysis

6 Analysis

Certain improvements to our model are more easily explained than others. In this section, we break down our analysis of model performance into two parts covering both the triplet loss function and Q&ABERT.

6.1 Triplet Loss Function

The modest yet comprehensive quantitative improvements outlined in table 2 and shown in figure 4a compared to figure 3a align with the behavior we empirically observed during training; namely, that the triplet loss started out low, even though it did improve through training. When training the BERT parameters in order to shape the distribution of embeddings as described in section 4.2.1, we observed Euclidean distances between anchor sentence embeddings and their corresponding positive sentence embeddings already tended to be quite low, often between 3 and 10. In contrast, distances between anchor embeddings and corresponding negative samples were often in excess of 15.

However, this was not a universal pattern. Exceptions to these observations occurred, in our rough approximation, in approximately 5% of cases. After training the shared BERT model for epoch 0, the triplet loss function evaluated to approximately 2.8. This value, already low, rapidly declined to approximately 0.03 after only 5 epochs. That our triplet loss started out so low yet still dropped substantially over the span of just a few epochs aligns with our experimental observations showing a modest yet notable improvement over baseline across all three tasks.

Our main interpretation of this behavior is twofold:

1. BERT was originally trained in such a way that produced a healthy distribution of CLS tokens throughout embedding space, leaving minimal improvements to be achieved via a naive embedding dispersal/clustering approach; and,

2. Our naive embedding dispersal/clustering approach is a relatively brute force tool for shaping embeddings that does not embody any substantial regard for the underlying semantics encoded in the embeddings.

6.2 Q&ABERT

Reframing the provided data sets as a question answering task led to improvements over baseline across all three tasks. Q&ABERT yielded significantly better results, with the most notable improvement occurring on the STS task.

At the time of writing, we are in a position only to hypothesize about reasons for this performance, as a full investigation requires time beyond that available for this project.

Viewed as a form of prompt engineering, Q&ABERT provides potentially valuable context to BERT and the task-specific layers with respect to the task at hand by reframing the input as questions. We speculate that reframing the provided data in this way may activate neural network pathways that have encoded deeper semantic word meanings through the attention mechanisms in the transformer architecture used by BERT. By providing additional context to the model, it appears to be much more capable of making sense of the inputs we are providing, compared to the baseline approach.

This seems potentially plausible (although again it is far from established) given the following insight. Recall that one of the two training objectives originally used to pretrain BERT was identifying whether one sentence follows logically from another. It is this training objective that primarily shapes the CLS token, which we use exclusively for all three tasks. For tasks like SST where only one sentence is present, it seems logical that CLS tokens would be less applicable since they are trained on the logical continuity between two sentences rather than analysis of individual sentences sans context. Therefore contextualizing the sentence to be analyzed as per Q&ABERT seems to make it a more natural fit with respect to at least one of BERT's original training objectives.

Although there are two sentences included in each of the STS and paraphrase inputs, neither of these scenarios are interchangeable with identifying semantic continuity between two sentences. We speculate that by providing additional context in the form of a question, the model is more effectively able to adapt to the specific task at hand rather than having to reason about two sentences with no guidance as to the goal of such reasoning.

Given additional time, we would have liked to explore attention saliency maps to investigate how the focus on keywords in the inputs varies between the baseline model and Q&ABERT.

7 Conclusion

We learned a tremendous amount from this project, spanning topics from theoretical NLP concepts and neural network architecture to PyTorch and MLOps tools. Our best achievement is the impressive performance yielded by Q&ABERT, and although triplet loss did not work out as well as we initially hoped, we gained a lot of insights from the experience implementing a custom loss function.

One key limitation of our work is our inability to substantively support our speculation about why Q&ABERT works as well as it does. We look forward to further investigating our hypothesis that attention mechanisms are involved in the improved expressiveness of the Q&ABERT model as future work.

References

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

[2] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The natural language decathlon: Multitask learning as question answering. *CoRR*, abs/1806.08730, 2018.

[3] Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*, 2017.

A Appendix

Table 3: Dataset classification proportions

		Classification				
		0	**1**	**2**	**3**	**4**
SST	Train	12.7%	26.0%	19.0%	27.2%	15.1%
	Dev	12.6%	26.2%	20.8%	25.3%	15.0%

		Paraphrase	**Non-paraphrase**
Quora	Train	63.1%	36.9%
	Dev	62.5%	37.5%

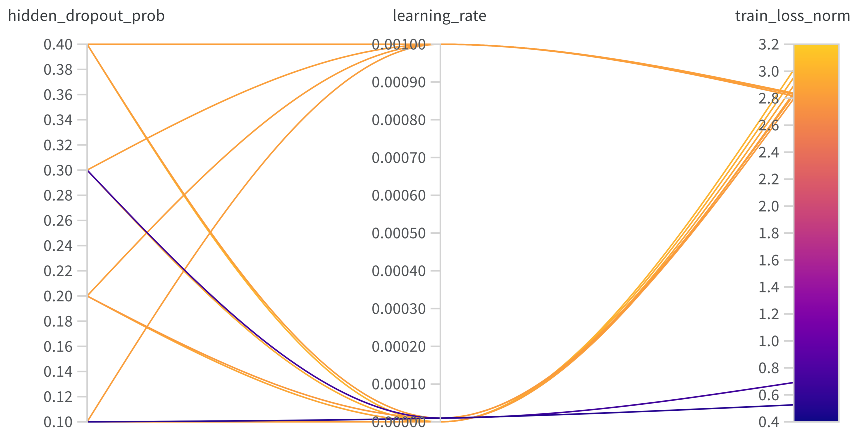


Figure 6: Results of hyperparameter grid search across dropout probability $\in [0.1, 0.2, 0.3, 0.4]$ and learning rate $\in [1e-3, 1e-5, 1e-7]$.