Improving Out-of-Domain Question Answering with Auxiliary Loss and Sequential Layer Unfreezing

Stanford CS224N Default Project (Robust QA Track)

Kai E. Okada
Department of Statistics
Stanford University
kaiokada@stanford.edu

Jason Z. Qin
Department of Chemical Engineering
Stanford University
jzqin@stanford.edu

Abstract

The proliferation of pretrained Language Models such as BERT [1] and T5 [2] has been a key development is Natural Language Processing (NLP) over the past several years. In this work, we adapt a DistilBERT [3] model, pretrained on masked language modeling (MLM), for the task of question answering (QA). We train the DistilBERT model on a set of in-domain data and finetune it on a smaller set of out-of-domain (OOD) data, with the goal of developing a model that generalizes well to new datasets. We significantly alter the baseline model by adapting an auxiliary language modeling loss, adding an additional DistilBERT layer, and undergoing training with sequential layer unfreezing. We find that adding an additional layer with sequential layer unfreezing offered the most improvement, producing a final model that achieved **EM = 42.23** and **F1 = 60.81** scores on the OOD test set.

1 Introduction

Within the field of Natural Language Processing (NLP), Question Answering (QA) is a challenging task often used as a benchmark for evaluating the quality of language models [4]. For QA tasks, models must read passages of text and answer questions about them. Specifically, given a context passage of text and a question to answer, QA models must choose a span of contiguous words within the passage that contains the answer to the question posed. This problem is challenging because models must accurately learn the meaning of both the context passage and the corresponding question, as well as the relationship between them.

Furthermore, QA models are sensitive to the type of data they are trained on. Models trained on QA data from one source will suffer from deteriorated performance when evaluated on QA data from other sources [5]. The goal of this work is to create models that have improved capabilities for predicting QA spans for out-of-domain (OOD) data.

This paper seeks to re-implement and improve upon models previously introduced for QA tasks. NLP models applied to specific tasks like QA often use pretrained language models as templates. These pretrained models are trained on large corpora for extensive periods of time, and serve as baselines from which finetuned models can be constructed. For this work, we start from a pretrained DistilBERT [3] model and finetune it for QA. We modify the DistilBERT model in several significant ways, incorporating ideas from a variety of related work. Specifically, we adapt an auxiliary masked language modeling (MLM) loss, add an additional transformer layer, and include sequential layer unfreezing from [6]. Our work differs from [6] because their improvements for the QA task were originally implemented on an LSTM model, whereas we adapt their ideas for a DistilBERT model. Furthermore, we experiment with span masking from [7] as the auxiliary loss, and additionally incorporate dynamic batch masking from [8]. In literature, both these techniques have been shown to improve performance on language modeling tasks.

We perform independent experiments to validate the role of incorporating an MLM loss, an additional transformer layer, sequential layer unfreezing, and a span masking loss. We find that these methods

improve upon a baseline DistilBERT model to different degrees, both separately and in combination. In addition, we analyze the outputs of these models to determine failure cases, and offer suggestions for further improvements.

2 Related Work

Transformer [9] models have significantly improved language model performance. By using self-attention as the primary method of relating different parts of textual input, transformer models effectively capture relationships between different parts of text, and have become popular tools for language modeling problems like QA. Furthermore, transformer models have been improving rapidly. Models like BERT [1], RoBERTa [8], and SpanBERT [7] improve transformer modeling incrementally by providing bidirectional language modeling, dynamic batch token masking, and span token masking, which have all been shown to improve language model performance. Likewise, DistilBERT [3] is a lightweight version of BERT trained with knowledge distillation. These rapid improvements in transformer architecture and training can be leveraged simultaneously to improve QA model performance.

Prior to transformer models, the most accurate language models were long short term memory (LSTM) networks [6]. One such model is SiATL [6], which applied pretrained LSTMs to OOD tasks. The creators of SiATL started with a pretrained LSTM, then incrementally improved it for various downstream tasks such as sentiment analysis and emotion detection. They improved upon the pretrained models by incorporating an auxiliary language modeling loss, adding an additional LSTM layer, and adopting sequential layer unfreezing. The auxiliary modeling loss was used to prevent the model from forgetting general features of language comprehension as it was finetuned on specific downstream tasks. The LSTM layer added parameters that the model could adjust for the new tasks. In addition, this new layer is used only for the target task rather than the auxiliary language modeling, thereby separating the target loss from the auxiliary loss and allowing the pretrained LSTM to retain general language modeling while also providing an addition to the architecture specific for the new application. Sequential layer unfreezing allowed the model to tune the new LSTM layer before adjusting the parameters and embedding layer of the pretrained model. Together, these improvements achieved a 5%-10% improvement over baseline models.

In our work, we seek to combine the relatively simple but effective improvements demonstrated by SiATL with recent enhancements in transformer architectures. Although the improvements provided in the SiATL paper were used on LSTM models, they extend naturally to transformer models as well. Furthermore, since recent transformer architectures have been shown to improve general language modeling, we hope that by combining their improvements with the modifications shown in SiATL, we can generate more accurate models for OOD tasks.

3 Approach

3.1 Baseline

We adapt a pretrained DistilBERT [3] language model for the QA task proposed in the RobustQA Default CS224N final project. As a baseline, we use the model provided by the class, and train it on a QA task on a large corpus of QA data (datasets described below). As an additional step of model adjustment, we finetune this model on a smaller set of OOD training data, before evaluating on unseen OOD data.

3.2 Improvements

Auxiliary Loss: To improve on these baselines, we adapt methodologies first proposed by Chronopoulou et al. in their SiATL model [6], by Liu et al. in RoBERTa [8], and by Joshi et al. in SpanBERT [7]. From SiATL, we adapted the use of an **auxiliary loss** during QA training. We experiment with several types of auxiliary loss. The first auxiliary loss we used is a masked language modeling (MLM) loss. For the MLM loss, we follow the masking scheme originally proposed in BERT [1], in which we randomly select 15% of all tokens, of which we mask 80%, change 10% to other random tokens, and leave 10% unchanged. To implement MLM masking, we adapt some masking code from [8], and implement it in our codebase. We also compare MLM loss to span

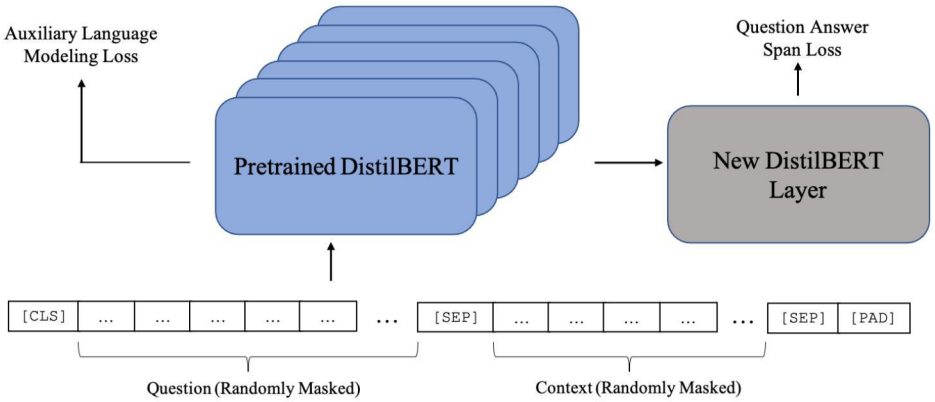

Figure 1: Architecture of modified DistilBERT model developed for this project. The input consists of a context paragraph followed by the question posed for the QA task. The pretraiend DistilBERT transformer has 6 layers, and is used to predict masked tokens for the auxiliary loss. The output of the hidden layer of the new DistilBERT layer is used to predict QA span tokens for the QA loss.

masking, in which we mask contiguous sequences ("spans") of text rather than individual tokens. It is hypothesized that span masks force the model to better learn semantic relationships over the entire input sequence, in order to fill in longer gaps of masked text. We experiment with different average span mask lengths, and follow the same distribution of masked vs. changed vs. unchanged tokens as presented above for MLM masking. To implement span masks, we develop our own code following the specifications provided in [7]. For both MLM and span masking, we only mask text tokens, and do not mask special tokens like [CLS], [SEP], and [PAD].

To incorporate the auxiliary loss, we follow the masking scheme proposed in RoBERTa [8]. RoBERTa differs from BERT in that all inputs are randomly re-masked at every epoch, instead of being masked once in pre-processing then used repeatedly over the training process. RoBERTa's masking approach has been shown to improve model prediction results [8]. The model then predicts QA tokens and MLM tokens simultaneously from these masked inputs. Our final model includes some masking code adapted from RoBERTa, as well as our own code for implementing a combined loss function. The loss functions take the form

$$L_{QA} = -\log \mathbf{p}_{start}(i) - \log \mathbf{p}_{end}(j)$$

$$L_{LM} = CrossEntropyLoss(\hat{y}_o, y_o) \text{ for } y_o \in \{\text{masked tokens}\}$$

$$L_{total} = L_{QA} + \gamma L_{MLM}$$

where γ is a hyperparameter that we decrease during training to bias the model toward focusing on QA loss in later epochs. During evaluation, only the QA loss function is used, and no token masking occurs.

Sequential Layer Unfreezing with Additional QA Layer:

Inspired by SiATL [6], we also add a new DistilBERT layer for QA modeling, and implement sequential layer unfreezing (SLU). For both these improvements, we implement our own modifications to the DistilBERT codebase. The new DistilBERT layer is appended to the end of the pretrained transformer layer, and is randomly initialized with Xavier initialization [10]. With SLU, we specify which layers can have their parameters updated during each training epoch. For the first N epochs, we only allow parameter updates for the new DistilBERT layer. After N epochs, we allow the entire model to train. The full architecture of our model is shown in Figure 1.

After training DistilBERT on the in-domain dataset with the combined loss model, we finetune the model on the limited OOD data, and then evaluate the model. We also attempt other experiments, in which we sequentially use MLM loss followed by QA loss (without any data masking during QA loss training). Full details are presented below. We compare the results of these models to the baseline QA model provided by the class.

4 Experiments

4.1 Data

The datasets we use are provided by the CS224N teaching staff. Each input consists of a context-question pair, separated by [SEP] tokens. The answer to each question is located within the context. The context-question inputs have a maximum length of 384 tokens.

The data are divided into two categories: QA in-domain and QA out-of-domain (OOD) data. Precise details about each dataset can be found from the original papers for each data source. The amount of data used from each data source is shown in the table below.

Dataset	Question Source	Passage Source	Train	Dev	Test
`in-domain datasets`					
SQuAD [4]	Crowdsourced	Wikipedia	86,588	10,507	-
NewsQA [11]	Crowdsourced	News articles	74,160	4,212	-
Natural Questions [12]	Search logs	Wikipedia	104,071	12,836	-
`out-of-domain datasets`					
DuoRC [13]	Crowdsourced	Movie reviews	127	126	1,248
RACE [14]	Teachers	Examinations	127	128	419
RelationExtraction [15]	Synthetic	Wikipedia	127	128	2,693

4.2 Evaluation Method

To assess the performance of our model, we use an evaluation method provided by the CS224N staff. The best performing models are determined by their exact match (EM) and F1 scores of QA span predictions. The EM score is a binary score that assesses whether the proposed start and stop tokens exactly match the true start and stop tokens for the data. The F1 score is a harmonic mean of the precision and recall for both the start and stop tokens. The models are evaluated on the OOD dev dataset, and the best model from this evaluation is selected for modeling on the OOD test dataset.

4.3 Experimental Details

General Implementation Details: Unless otherwise specified, the following implementation details and hyperparameters apply for all models listed below. Models were trained on in-domain data for 3 epochs, then finetuned on OOD data for 10 epochs. Models used the Adam optimizer with a learning rate of 3e-5, and were trained with batch size = 16. During in-domain training, models were evaluated on the in-domain dev set every 2000 training steps, and the best-performing model was chosen as the starting model for OOD training. During OOD training, models were evaluated on the OOD dev set every 100 steps, and the best-performing model was chosen as the "final" model for each experiment. The baseline architecture used for this project, and upon which all modifications were made, is a 6-layer DistilBERT model adapted from HuggingFace [16].

Baselines: We begin with a pretrained `DistilBertForQuestionAnswering` [17] model. After training on the in-domain data, it is 1) directly evaluated on the OOD dev data (yielding the Baseline model), and 2) further finetuned on the OOD training data for 10 epochs (yielding model M_1).

Stagewise Losses: As an intermediate step toward our auxiliary loss model, we first discretely finetune the "pretrained-base" DistilBERT using `DistilBertForMaskedLM` [17] on the in-domain dataset for one epoch, then transfer the weights to a `DistilBertForQuestionAnswering` model and continue fine-tuning for 3 epochs on the QA task. Finally, we further finetune on the OOD dataset using only QA loss (yielding model M_2).

Auxiliary Masked Language Modeling Loss: Drawing from the DistilBERT Hugging Face codebase [16], we implement our own auxiliary loss model that maintains masked language modeling (MLM) loss throughout in-domain training. When training with the MLM loss, we mask according to the scheme proposed in BERT [1], and described above. We keep always keep 3 tokens unchanged in the inputs: `[CLS]`, `[SEP]`, and `[PAD]`. We keep `[CLS]` and `[SEP]` tokens unchanged because we want to ensure the model learns the relationship between the question and context strings. We leave `[PAD]` unchanged since it has no semantic relationship to each input.

Because we use masking for the auxiliary loss on in-domain training, the inputs are also masked for the QA loss. This likely increases the difficulty for the model to converge to a minimal QA loss, but we hope that this decrease in QA accuracy is offset by an increase in general language modeling via the auxiliary loss. When using the auxiliary loss, the combined model loss function is $L_{total} = L_{QA} + \gamma L_{MLM}$. In our experiments, we gradually decrease the γ as we train the model. We modulate both the upper and lower limits of γ, as well as the rate at which it decreases.

For OOD training, we do not mask any inputs, nor use an auxiliary loss. As such, the OOD training is only run on QA loss. The final model for this experiment, after OOD finetuning, is labeled M_3.

Span-Masking Auxiliary Loss: We found our experiments using an MLM loss to be successful, and decided to try alternative auxiliary losses. Given the success of SPANBert [7], we create a span-masking variant of M_3. Following the methodology in SPANBert, we draw span lengths $l \sim \text{Geom}(p)$, i.i.d., with $p = 0.20$, and clip these lengths to a given maximum length (l_{max}). We then select a subset of the spans so as to mask no more than 15% of the input sequence plus the expected span length prior to clipping, and assign each span to a starting index drawn uniformly from the input. Given that span prediction may pose a more difficult task than individual token prediction, we experiment with varying l_{max} for in-domain training (M_4 and M_5), with $l_{max} = 1$ assumed to correspond with our original auxiliary MLM model. These results are shown in Table 5.

Sequential Layer Unfreezing (SLU) with Additional QA Layer: To implement sequentual layer unfreezing (SLU), we first add an additional transformer layer to the pretrained DistilBERT model. This layer is added to the end of the pretrained DistilBERT model, and takes as input the final hidden state of the last layer of the pretrained DistilBERT model. The output of this layer is passed to a feedforward network (FFN) that predicts the QA span tokens. The new layer DistilBERT layer is initialized randomly with Xavier initialization.

To investigate the effect of adding a new layer, we perform several experiments. First, we wanted to see whether the larger model could train as well as the 6-layer model on the same number of training steps. To test this, we allowed the combined model to train in-domain for 3 epochs, followed by OOD finetuning, yielding M_6.

Next, we introduce sequential layer unfreezing. For the first 2 epochs of in-domain training, we only allow parameter updates to the newly initialized DistilBERT layer and QA FFN, while keeping the pretrained layers frozen. Because the new DistilBERT layer is randomly initialized, we use the first 2 epochs to allow this layer to converge to a local minimum in the loss landscape. At this point, we consider the new layer "trained", and then we train the entire model for 3 epochs, producing model M_7. This experiment allows us to compare how a 7-layer DistilBERT transformer to a 6-layer DistilBERT model (i.e. the baseline) when both are entirely unfrozen for 3 epochs of training.

Combining Methods: We take our best auxiliary loss regimes based on validation set performance (M_4 and M_5) and combine them with SLU (M_7) over 5 epochs (yielding M_8 and M_9). We train on the in-domain dataset for $N = 2$ epochs with the base layers frozen (and no masking or auxiliary loss), then unfreeze all layers and train for 3 epochs with masking and an auxiliary loss. We again decay γ linearly from 2.0 to 0.5 for in-domain training over the 3 epochs of full-model training.

Notably, when we incorporate the auxiliary loss with the SLU + extra layer architecture, we follow the scheme proposed in SiATL, and allow the additional layer to only predict the QA loss. More specifically, the auxiliary loss is predicted by the output of the pretrained, 6-layer model. The last hidden state of the 6-layer model is used as input to the new DistilBERT layer, and the output of this new layer is used to predict the QA span. The new layer does not impact the prediction of masked tokens. We hypothesize that separating the losses in this fashion allows the first 6 layers to maintain a general understanding of language, while allowing the new layer to carefully attend to the QA task.

4.4 Results

Based on evaluation upon the OOD dev set (Table 1, the best performing models were models that incorporated an MLM auxiliary loss on in-domain modeling (M_3, **F1 = 50.35, EM = 34.81**), and the SLU model that underwent $N = 2$ epochs of frozen training, followed by 3 epochs of full-model training (M_7, **F1 = 50.36, EM = 35.34**).

Method	Description	EM (dev)	F1 (dev)
Baseline	In-Domain QA only (3 epochs)	31.68	47.10
M_1	In-Domain QA (3) + OOD QA (10)	**34.81**	**48.50**
M_2	In-Domain MLM (1) + In-Domain QA (3) + OOD QA (10)	32.72	48.43
M_3	In-Domain [QA + MLM] (3) + OOD QA (10)	**35.60**	**50.35**
M_4	In-Domain [QA + SM l_{max} = 2] (3) + OOD QA (10)	34.29	49.59
M_5	In-Domain [QA + SM l_{max} = 4] (3) + OOD QA (10)	34.29	48.32
M_6	In-Domain [SLU $N = 0$, no Aux] (3) + OOD QA (10)	35.08	48.80
M_7	In-Domain [SLU $N = 1$, no Aux] (5) + OOD QA (10)	35.34	**50.36**
M_8	(M_3 + M_7) + OOD QA (10)	33.51	48.72
M_9	(M_4 + M_7) + OOD QA (10)	34.82	49.17

Table 1: **Model Iteration**. Models M_1 - M_9 represent the addition of auxiliary loss and sequential layer unfreezing to our baseline, both individually and in combination. All scores are based on the OOD dev set. SM = Span Masking, Aux = Auxiliary Loss

Comparing the Baseline model with M_1, we see that finetuning the model on OOD data significantly improves the both F1 and EM performance. As such, we incorporate OOD QA finetuning on all downstream models. This conclusion is supported by additional results comparing the performance of models on the OOD dev set before and after OOD finetuning (Tables 4 and 5 located in the **Appendix**). In most cases, incorporating OOD finetuning increased EM and F1 scores by 10% - 20% compared to only in-domain training.

Furthermore, comparing M_3 with M_1 convinced us that incorporating an auxiliary loss offers significant improvement on QA performance. Given the success of the MLM loss, we decided to incorporate a span loss, the results of which are shown in models M_4 and M_5. Unfortunately, the span loss models did not seem to perform as well as the MLM model, with results shown in Table 5 located in the Appendix. We experimented with maximum span lengths of $l = 1$, 2, and 4, and found that the model with l=1 (i.e. MLM) performed best. We hypothesize that this is because the auxiliary span-masking loss differs significantly from the MLM loss upon which the DistilBERT models were pretrained. The decreasing performance of models with longer span masks is shown in 5. While SPANBert sets $l_{max} = 10$ [7], it specifically pretrains on this objective over a much larger quantity of data than our in-domain training set.

In addition, comparing M_7 (a 7-layer model with 3 epochs of full training) to M_1 (a 6-layer model with 3 epochs of full training) demonstrated to us that including an extra layer with sequential layer unfreezing also provides significant improvement on the QA task. In our final tests, we combined our best auxiliary loss models with our best SLU models, yielding M_8 and M_9. Although these final models incorporated benefits from both auxiliary loss models and SLU models, both M_8 and M_9 performed more poorly than either the best auxiliary loss models or best SLU models. In fact, M_8 and M_9 are comparable in performance M_1, which is simply the baseline model with OOD finetuning.

Since models M_3 and M_6 performed best, we submitted those models for evaluation on the OOD test set. In addition, we submitted model M_9 to investigate how the combined model performed on the OOD test set, and M_1 to determine the performance of the baseline model. The performance of these models on the test set is shown in Table 2.

This table shows that based on F1 scores on the OOD test data, **the SLU model without an auxiliary loss (M_7) performed best, with F1=60.81 and EM=42.23**. Surprisingly, M_1, which is the baseline model of an unmodified DistilBERT model trained on both in-domain and OOD data, performed best based on EM scores, and also had the second best F1 score. We evaluated M_9 twice on the test data, 1) without any layer unfreezing during finetuning on the OOD data (model M_9), 2) with one epoch of training with only the additional DistilBERT layer unfrozen, followed by 2 epochs of full model training (model M_9'). M_9' outperformed M_9, but failed to match performance of M_7. This suggests that we were unable to successfully merge the best independent models together. Furthermore, that all models beside M_7 performed strictly worse than M_1 suggests that including an auxiliary loss does not improve model performance on the test set. We discuss possible reasons for failure below.

Method	EM (test)	F1 (test)
M_1	**42.317**	59.452
M_3	41.078	58.170
M_7	42.225	**60.807**
M_9 (9 epochs OOD)	41.170	57.876
M_9' (3 epochs SLU OOD, $N = 1$)	41.766	58.587

Table 2: **Test Submissions**. EM and F1 scores of models making predictions on QA OOD test data.

5 Analysis

5.1 Error Analysis by Data Source and Question Type

To understand why M_7 scored particularly highly on the test set despite achieving lower scores than M_3 on the dev set, we examine dev set performance on particular data sources and question types (e.g. questions starting with "Who...", "What...", etc.). These results are shown below in Table 3.

	% of Dev Set	M_1	M_3	M_7	M_9
All Data	100%	31.94%	**32.72%**	30.37%	31.41%
DuoRC	32.98%	25.40%	26.98%	**30.16%**	25.40%
RACE	33.51%	**20.31%**	18.75%	17.97%	17.19%
RelationExtraction	33.51%	50.00%	**52.34%**	42.97%	51.56%
"Who..."	23.56%	25.56%	27.78%	**33.33%**	30.00%
"What..."	40.84%	34.62%	**37.18%**	31.41%	33.97%
"Which..."	12.04%	**41.30%**	26.09%	26.09%	26.09%
"Where..."	6.28%	16.67%	**33.33%**	20.83%	16.67%

Table 3: **Model Accuracy on Different Data Sources and Question Types.** Percent accuracy was determined by computing EM scores for comparing the model's output to dev set answers, removing all punctuation. "When", "Why", and "How" questions are excluded due to scarcity in the dataset.

Relative to our baseline model finetuned on the OOD training dataset (M_1), all of our other models except for M_7 improve the model's performance on the RelationExtraction dataset. Interestingly, M_7, which obtained the best test set performance of all of our submitted models, only achieves the best dev set accuracy on "Who" questions and the DuoRC dataset. While all data sources are represented evenly in the OOD training and dev datasets, the RelationExtraction and DuoRC datasets feature much more prominently in the OOD test dataset. Furthermore, when we examine the test data, we find that "Who" questions comprise 51% in the DuoRC dataset at, but feature at approximately 15% in other datasets. The auxiliary masking loss models appear to favor performance on RelationExtraction and "What" questions, and it is possible that relatively low DuoRC and "Who" results may have hurt their overall performance. We note some specific DuoRC examples in Table 7, in the **Appendix**.

In particular, we observe a common pattern in many of our models in which the output answer span for certain questions is very large, even though the tokens at the ends of the span appear to reflect either the same answer or two conflicting responses. This indicates a sense of "indecisiveness" - the model has not fully learned that its start and end tokens should always encompass a single coherent response. M_7 makes this type of mistake on "Who" questions in the dev dataset only 6.67% of the time, whereas this figure is at least 14% for both M_3 and M_9. In general, the DuoRC dataset is unique in that it supplies a large number of question and answer pairs for a small number of long contexts. It is possible that the additional transformer layer helps the model retain more contextual information over the longer passage, while the auxiliary loss finetuning in M_9 detracts from the process of building this contextual representation. This may have given M_7 a comparative advantage in test set performance. While its performance on the RelationExtraction dataset appears to decline in Table 3, this is largely explained by the observation that M_7 prepends articles ("a", "the") to its answers at a much higher rate than the other models (for example, 5 times higher than M_3). This may be further evidence that M_7 preserves a more general grasp of context than the other models.

5.2 Out-of-Domain Fine-Tuning and Test Set Performance

All of our models M_1-M_9 were obtained after 10 epochs of OOD finetuning. We initially believed this additional step was beneficial given the results of M_1 and M_3 on the dev set compared to the baseline. However, the final M_7 model was chosen after only a single training step on the OOD training data, with the model decreasing in performance at later stages of finetuning. This suggests that additional training on the OOD dataset causes models to overfit to the OOD training data. Furthermore, it suggests that additional OOD finetuning is not strictly beneficial for model performance. As we selected models based on peak performance on the OOD dev set at regular intervals during OOD training, the final models had different levels of exposure to the OOD training set, and it is possible that this discrepancy resulted in some of our models exploiting correlations between training data and the dev set that do not generalize. The step and epoch of OOD finetuning at which each final model was chosen is shown in Table 6 of the **Appendix**. We believe this is also one possible reason why M_1 outperformed every model except M_7 on the OOD test data set, despite having lower EM and F1 scores on the OOD dev set. Since all models except M_1 had greater complexity than M_1, it is possible that performance of models M_3 and M_9 on the OOD dev set simply meant that they were overfit to common aspects of the OOD training and dev data.

Interestingly, the models that were trained with an auxiliary loss usually had relatively poor performance on the OOD dev set at the start of the OOD finetuning phase, but improved significantly in performance as OOD finetuning progressed (M_3, M_5). This suggests that these models may be able to gain more from the small OOD training set before overfitting relative to the dev set and suffering a decline in performance. However, the additional time required to attune these models to the OOD task also increases the risk of accidental overfitting relative to the test set. There appears to be a delicate balance between leveraging the auxiliary objective to increase the model's potential to learn from the small OOD sample and possibly overfitting over too many training epochs. As evidence of this, we found that regularizing M_9 by using SLU on the OOD fine-tuning step over only 3 epochs (M_9') yielded a better test set result despite comparable peak performance on the dev set (Table 2). It is also possible that lowering the range of γ or training for more epochs over the same range would improve the performance of our auxiliary models, as our M_3 model achieved its best single-shot OOD performance when finetuned in-domain for 4 epochs with γ decreasing from 2.0 to 0.5 (see Table 4 in the **Appendix**).

6 Conclusion

For the task of QA on an OOD dataset, we have developed a model that improves over a pretrained DistilBERT model. We sought to achieve improvements over the baseline by implementing the following suggestions proposed in SiATL [6]: implementing an auxiliary loss function, adding an extra DistilBERT layer, and training with sequential layer unfreezing. To mask inputs with the MLM model, we incorporate a dynamic masking scheme proposed in RoBERTa [8]. In addition, we experiment with using a span mask instead of MLM, but find that an auxiliary MLM loss outperforms auxiliary span losses. Although all the changes above improved our models over a baseline model based on the OOD dev dataset, our best performing model on the test set used only an extra DistilBERT layer with sequential layer unfreezing. This model achieved **EM = 42.23** and **F1 = 60.81**, placing us in the top 10% of the CS224N RobustQA Test leaderboard based on F1, and the top 30% based on EM.

Interestingly, when evaluating our final models on the OOD test data, we found that an unmodifed DistilBERT model with 3 epochs of in-domain and 10 epochs of OOD training outperformed all other models except our best one. This suggests that adding an auxiliary loss improves performance on the OOD dev data without necessarily improving performance on the OOD test data. In addition, combining the auxiliary loss model with the additional layer+SLU also resulted in a model that performed worse than the additional layer+SLU model. In both cases, we believe this is because of insufficient tuning of auxiliary loss hyperparameters, resulting in an overfit on OOD training and dev data. Going forward, we would improve our models by performing additional hyperparameter tuning. For instance, we would to perform more experiments about varying the weight of the auxiliary loss relative to the QA loss. We would also seek to improve the reliability of the dev dataset results via methods such as random sampling or bootstrapping. Furthermore, since we believe that span masking as an auxiliary loss was less successful than MLM because of an insufficient amount of training, we would like to repeat our span masking experiments with many more epochs of in-domain training.

7 Acknowledgements

Overall, we found this project to be a great learning experience, and thank the teaching staff for the hard work they put into setting up the default project. We look forward to using the skills we have learned and implementing them in practice!

References

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

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

[3] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *arXiv preprint arXiv:1910.01108*, 2019.

[4] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In *CoRR, abs/1606.05250*, 2016.

[5] Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. MRQA 2019 shared task: Evaluating generalization in reading comprehension. *CoRR*, abs/1910.09753, 2019.

[6] Alexandra Chronopoulou, Christos Baziotis, and Alexandros Potamianos. An embarrassingly simple approach for transfer learning from pretrained language models. In *Association for Computational Linguistics (ACL)*, 2019.

[7] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. Spanbert: Improving pre-training by representing and predicting spans, 2020.

[8] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. In *arXiv preprint arXiv:1907.11692*, 2019.

[9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. JMLR Workshop and Conference Proceedings.

[11] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. Newsqa: A machine comprehension dataset. In *Association for Computational Linguistics (ACL)*, 2017.

[12] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, , and Slav Petrov. Natural questions: a benchmark for question answering research. In *Association for Computational Linguistics (ACL)*, 2019.

[13] Amrita Saha, Rahul Aralikatte, Mitesh M. Khapra, , and Karthik Sankaranarayanan. Duorc: Towards complex language understanding with paraphrased reading comprehension. In *Association for Computational Linguistics (ACL)*, 2018.

[14] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, , and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

[15] Omer Levy, Minjoon Seo, Eunsol Choi, , and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension. In *Association for Computational Linguistics (ACL)*, 2017.

[16] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

[17] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, October 2020.

8 Appendix

Method	EM (ID)	EM (ID + OOD)	F1 (ID)	F1 (ID + OOD)
M_3 - 2 epoch ID	31.68	34.03	46.42	49.32
M_3 - 3 epoch ID	29.58	**35.60**	46.19	**50.35**
M_3 - 4 epoch ID	**34.55**	34.82	**49.59**	49.76

Table 4: **Impact of Number of Epochs of In-Domain Fine-tuning with Auxiliary Loss on Dev Set Performance**. We compare zero-shot and few-shot performance on the OOD dev dataset ("ID" = In-Domain fine-tuning only, "ID + OOD" = both).

Method	l_{max}	EM (ID)	EM (ID + OOD)	F1 (ID)	F1 (ID + OOD)
Baseline / M_1	0	31.68	34.81	47.10	48.50
M_3	1	29.58	**35.60**	46.19	**50.35**
M_4	2	32.20	34.29	47.74	49.59
M_5	4	31.41	34.29	46.01	48.32

Table 5: **Impact of Masked Span Length on Dev Set Performance**. We use $l_{max} = 0$ to refer to no masking and $l_{max} = 1$ to refer to random masking. We compare zero-shot and few-shot performance on the OOD dev dataset ("ID" = In-Domain fine-tuning only, "ID + OOD" = both).

Method	One-shot EM (dev)	One-shot F1 (dev)	OOD Peak Training Step
M_1	32.46	47.44	100 (Epoch 2)
M_2	31.41	47.23	100 (Epoch 2)
M_3	29.84	46.32	400 (Epoch 8)
M_4	32.46	47.14	300 (Epoch 6)
M_5	31.41	46.15	100 (Epoch 2)
M_6	34.03	46.71	750 (Epoch 15)
M_7	35.34	50.19	0 (Epoch 0)
M_8	29.06	46.83	600 (Epoch 12)
M_9	32.98	47.29	400 (Epoch 8)
M_9'	33.25	47.27	140 (Epoch 2)

Table 6: **Performance at Start of OOD Fine-Tuning and Peak Location**. "Peak Training Step" refers to the training step at which the best model was evaluated against the dev dataset and saved.

Question	**Correct Answer**	M_3	M_7	M_9
Who is Harry's travelling companion?	his old friend Jacob	Ginger continues west with Harry. She claims she is 16, and is running away from home to a commune in Boulder.Harry and Ginger	Tonto	Annie died.Harry picks up two young hitchhikers, one of whom soon gets another ride, while the other a girl named Ginger
Who seeks revenge from Chaco?	Stubby	Stubby and Bunny	Stubby	The arrive in a small town in the pouring rain, but discover that it's a ghost town. The following morning is bright and sunny. Bud
What is the name of the place where Young Man with a Skull can be found?	National Gallery	National Gallery	the National Gallery, London	National Gallery

Table 7: **Sample Answer Spans.** Examples of answers proposed by the models M_3, M_7, and M_9 on examples from the DuoRC (first two) and RelationExtraction (third) datasets.