Robust QA System with Task-Adaptive Pretraining, Data Augmentation, and Hyperparameter Tuning

Stanford CS224N Default Project: Robust QA Track

Mentor: Alvin Hou

Emily You
MS&E
Stanford University
emilyyou@stanford.edu

Kun Qian
MS&E
Stanford University
donaqian@stanford.edu

Xingzi Xu
MS&E
Stanford University
xingzix@stanford.edu

Abstract

In this project, we aim to build a robust question answering (QA) system by improving the DistilBERT model[10]. A robust system eliminates the need for prohibitively large datasets in training models for use in different domains. To accomplish this goal, we implement task-adaptive pretraining (TAPT), model tuning such as transformer block re-initialization and increasing the number of training epochs, and ensemble methods. Since there is a limited amount of out-of-domain data, we also use data augmentation techniques for both pretraining and finetuning steps. Our final ensemble model achieves dev scores of F1 = 52.57, EM = 37.96 and test scores of F1 = 60.42, EM = 43.42.

1 Introduction

Despite the tremendous progress in natural language processing (NLP) modeling over the last decade, one remaining challenge in the field is in creating NLP systems that can generalize like humans. There have been a number of researches on models that can perform decently well by learning superficial correlations. However, these models often fail to perform well on out-of-distribution data [1] [2] [3] [4]. The ability to generalize is crucial for the question answering task because training and test data often come from different distributions in real life. This robustness is fundamental to NLP as it indicates how well machine learning models "understand" the input text.

We aim to build a robust QA system that can utilize information learned from pretraining data, adapt to unknown domains with only a few training samples, and produce meaningful answers. In this project, we are given a large amount of in-domain data and very limited out-of-domain training data. Our final goal is to construct a robust QA model that can perform well on the out-of-domain test data. We explore three methods to accomplish this: task-adaptive pretraining (TAPT), out-of-domain data augmentation, and model tuning techniques such as re-initialization, number of epochs tuning, and ensemble modeling. After investigating whether each method improves the performance on its own, we combine the beneficial methods together to build the most robust model.

2 Related Work

One straightforward method to enhance a question answering model's robustness is to let the model gain more knowledge on the domains of our interest. The paper "Don't Stop Pretraining"[5] suggests TAPT, pretraining on domain or task-specific data before finetuning, to make models learn to do well on specific domains or tasks. Other studies have also shown that the performance of models can be enhanced by using text from target domains during this pretraining step, too. [6] In this paper, we apply this method by continuing to pretrain on task-specific data before finetuning in an effort to improve our model's performance on out-of-domain data.

Another big challenge in this project is that we are given very limited out-of-domain training data. In order to fully take advantage of the limited available data, we utilize data augmentation as explored by many other studies. For example, word substitution-based data augmentation replaces a randomly selected word from a sentence with a synonym from a lexicon [7], a word with similar word embeddings, [8], or [MASK] tokens which are then filled by BERT to produce augmented data. Another common way to augment text data is back translation, which translates a sentence to another language and then translates it back to the original language. [9].

Lastly, many recent works have shown the importance of hyperparameter tuning on fewshot performance. [16] An example is the re-initialization of top transformer blocks before fine-tuning. This idea is motivated by object recognition transfer learning results showing that lower pretrained layers learn more general features while higher layers closer to the output specialize more to the pretraining tasks. The authors claim that re-initialization consistently improves mean performance on few-sample datasets. Another hyperparameter is the number of training iterations. It is shown that training longer tends to improve model performance compared to the common practice of training for 3 epochs. Still, the paper does not explore the effects of re-initialization and training iterations on larger datasets, DistilBERT-based models, or TAPT models. In this paper, we experiment with the two hyperparameters to investigate these unexplored effects as we tune the models.

3 Approach

Baseline The baseline of this project is the DistilBertForQuestionAnswering model fine-tuned on in-domain training data as described in the project handout. The baseline obtains F1: 48.43 and EM: 33.25 on the out-of-domain validation data.

3.1 Task-Adaptive Pretraining (TAPT)

Compared to DistilBert[10], a DistilBertForQuestionAnswering model has an extra Question-Answering (QA)-specific layer, which is a linear layer on top of DistilBert backbone layers that generates start and end logits. Similarly, a DistilBertForMaskedLM model, used in TAPT, is a DistilBert model with an extra masked-language-modeling (MLM) head on top.

Inspired by the paper "Don't Stop Pretraining"[5], we adopt TAPT via continued pretraining with MLM heads. We take a model, initialize MLM heads, train with unlabeled, task-relevant corpus, and convert the resulting model into a QA-model structure. This completes the TAPT continued pretraining process (written simply as "pretraining" from now on) during which the model should be able to learn more out-of-domain knowledge. The MLM objective allows the model to effectively augment the pretraining data sets, as mentioned in the paper. All models are pretrained on unlabeled out-of-domain training data, which is generated by extracting and tokenizing all contexts, questions, and answers. The code for generating this unlabeled data in our original script, `tapt.py`.

We test two ways of implementing TAPT: with and without the baseline model. We first try TAPT on the baseline model to see if it boosts the performance. Next, we apply it to the DistilBert model. Then we compare the performance of all pretrained models on out-of-domain validation data and pick the best one to be finetuned later.

TAPT with the baseline model To utilize the in-domain knowledge learned by the baseline model during finetuning, we build a new pretrained model on the baseline model. This original approach is shown in Appendix A. We first initialize a DistilBertForMaskedLM model, which is created by taking the backbone layers (`self.distilbert`) from the baseline and initializing MLM heads. Then, this MLM model is trained with out-of-domain training data. Then, we load the shared weights (backbones) from the now trained DistilBertForMaskedLM model, stored in `self.distilbert`, to our baseline model. In this step, the top QA-specific layers from the baseline model are essentially stacked on top of `self.distilbert`. Then, this final model is fine-tuned with out-of-domain data. The resulting model is written as `Pretrain_Baseline_oodomain + Finetune_oodomain` in Appendix B. Since this resulting model does not outperform the baseline model, we stop exploring this TAPT method and switch to the TAPT on DistilBert approach.

TAPT without the baseline model TAPT without the baseline model only has two main differences from the previous approach: (1) the QA-specific layers are initialized randomly instead of coming from the baseline model, and (2) augmented training data are used in both pretraining and finetuning. Augmented data is used here because there is very little out-of-domain training data. In summary, this TAPT approach consists of: initializing MLM head on DistilBert, training it with augmented unlabeled out-of-domain data, taking its backbone and combining with initialized QA-layers, and fine-tuning with augmented labeled out-of-domain data. Note that finetuning was done for comparison purposes only - the pretrained model consists of all steps up until finetuning. We decide to use this approach in our final model. The general training scheme can be found in Figure 1 below.

For this part, we use `Finetune_augoodomain (back translation)`, an uncased DistilBert-ForQuestionAnswering model finetuned with augmented out-of-domain data, as a benchmark to examine the effect of TAPT.

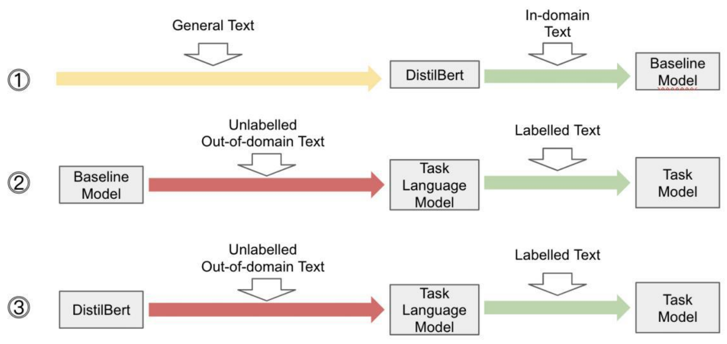


Figure 1: Training schemes. (1) Baseline model: one pretraining step (yellow arrow) with a large amount of general text (DistilBert) and one finetuning step (green arrow) on in-domain data, (2) `Section 4.1.1`: one TAPT step (red arrow) on baseline model with unlabeled out-of-domain data and one finetuning step on labeled out-of-domain data, (3) `Section 4.1.2`: one TAPT step on uncased DistilBERT model with out-of-domain data and one finetuning step with augmented out-of-domain data

3.2 Data Augmentation on Limited Out-of-domain Train Set

One challenge in this project is the limited amount of out-of-domain training datasets, DuoRc, RACE, and RelationExtraction, each with about 127 question-and-answer pairs. To cope with the difficulty of limited data, we explore data augmentation in two places. First, for pretraining MLM models, we augment the context and question (unlabeled) data from the original out-of-domain training sets. Since each context instance is very long, we break them into shorter sentences, which are then augmented to 10 folds the size of the split sentences. Thus, the model can learn more general language information through TAPT, specifically in tasks DuoRc, RACE, and RelationExtraction. Second, to obtain a larger out-of-domain training set for finetuning, we augment questions to 10 folds and label them with the correct answers to the original questions. For finetuning training data, we only augment the questions because changing the context or the answer will change the resulting answer from the model.

We explore four data augmentation techniques in NLP: 1) word replacement by similar embeddings using Word2Vec 2) word replacement by a formal synonym model like WordNet 3) synonym replacement from Easy Data Augmentation techniques (EDA SR), which is based on WordNet with more randomness in implementation 2); 4) back translation techniques built on top of the Google Translation API. All implementations are from the library `textaugment`. [12] [8] We do not use random swap (RS), random deletion (RD), and random insertion (RI) from EDA as they would change the questions drastically. Furthermore, the back translation method translates each question to one of the top ten high-resource languages and then translates it back to English.

From initial observation, we see that Word2Vec and back translation methods are time-consuming. We also observe that Word2Vec tends to produce nonsensical sentences. Considering running time, we are left with EDA SR and WordNet. Because EDA SR has more randomness and thus produces more unique augmented sentences, we only use EDA SR for pretraining augmentation. We use all four methods in the finetuning stage to compare their performances.

Data augmentation is implemented in `aug_context.py` and `aug.py`. Contexts are augmented with the EDA SR technique. After removing duplicated results, we obtain 38,660, 22,684, and 1,415 context sentences for DuoRC, RACE, and RelationExtraction, respectively. We combine these augmented contexts with augmented questions into a giant unlabeled dataset for TAPT. For questions, they are augmented and assigned the correct answer label. In order to avoid having many same augmented sentences, we introduce different probability thresholds to produce augmented sentences when using Word2Vec, WordNet, and EDA SR. With augmentation, we obtain over 1,000 question-and-answer pairs for the finetuning out-of-domain training data.

3.3 Hyperparameter Tuning & Ensemble Method

The main hyperparameters we experiment with are the number of re-initialized blocks and the number of training epochs. For re-initialization, the parameters of the top N transformer blocks are replaced with weights from the original DistilBERT model, which follow a normal distribution with 0 mean and standard deviation of 0.02. The number of epochs is also varied. Models are on default run with 3 epochs. All of the hyperparameters are evaluated on the out-of-domain validation sets assuming that they are good representations of the test data sets.

In the final step, we create two different ensemble methods to fully take advantage of the prediction power of the models we built. First, in the "Simple" approach, we create ensemble predictions from three different models based on the probabilities they output for each prediction. If the predictions from all of the three models match, then we are confident that they generate the correct answer and keep the prediction. Otherwise, the method selects the prediction with the highest probability, computed from taking the softmax of all scores (implemented in modified function postprocess_qa_predictions in `util.py`). The "Weighted Average" ensemble method takes the start and end logits of three different models and takes their weighted average. The most weight is placed on the model with the highest F1 score and the least weight on the lowest one. The code for this portion is written from scratch (`ensemble.py, ensemble_weighted.py`).

3.4 Overall Model Structure

To summarize the steps, a "TAPT-model" consists of TAPT applied to DistilBert model, which is then finetuned twice. First, the pretrained model is finetuned on in-domain training data so that it can learn general language distributions. Then, it goes through a second-stage finetuning on augmented out-of-domain training data to expose it to out-of-domain distributions. Various models like "TAPT-models" with different data augmentation techniques and models with different hyperparameters are combined through the ensemble method to generate the final model.

4 Experiments

4.1 Data

In this project, we are provided with three in-domain reading comprehension datasets and three out-of-domain datasets. For the baseline, we train the model using the in-domain datasets: Stanford Question Answering Dataset(SQuAD)[13], Natural Questions[14], and NewsQA[15], each with 50,000 examples. For TAPT, we use the augmented unlabeled out-of-domain datasets: DuoRC ($\sim$ 40,000), RACE($\sim$ 23,700), and RelationExtraction ($\sim$ 2,400). (The numbers are not exact due to the randomness in the data augmentation algorithms.) For finetuning, we use augmented labeled out-of-domain training set from the four data augmentation techniques and compare their impact on the dev scores. For evaluation of each model, we use the out-of-domain validation sets, each with about 128 points. Lastly, three final candidate models are assessed on the out-of-domain test sets to get unbiased estimates of their performances.

4.2 Evaluation method

EM (Exact Match) and F1 scores across the entire out-of-domain validation set are used as metrics to compare the performance of the created models against the baseline model. The F1 score balances precision and recall by taking their harmonic mean. EM determines whether the produced output is an exact match to the correct label.

4.3 Experimental details

Task-Adaptive Pretraining (TAPT) The setup of our experiments for the two TAPT methods are shown in Figure 1. Each TAPT is performed with 3 epochs over the unlabelled training text, a learning rate of 1e-4 and a masking probability of 0.15, which is proposed by Gururangan et al. [5]. For TAPT without baseline, we try two augmented out-of-domain texts for pretraining: (1) questions augmented 10-fold with back translation, and (2) contexts and questions augmented with EDA 10-fold. For finetuning, 10-fold question-augmented out-of-domain text with back translation is used. This experiment helps us find the best pretrained model to use for the next steps.

Data Augmentation Technique Selection in Finetuning In this section, we want to answer two questions: 1) Does text data augmentation improve performance? 2) If so, which augmentation technique gives the best performance? Specifically, we finetune the baseline model using the original out-of-domain training set and the augmented out-of-domain training sets obtained from the four augmentation techniques. The results are shown in Table 1. We then run the same set of experiments on the top of the TAPT pretrained models, shown in Table 2. For each technique, the out-of-domain data used during the second finetuning stage is augmented. All the experiments use the default learning rate (3e-5), number of epochs (3), batch size (16), etc.

Hyperparameter Tuning & Ensembling Method Hyperparameter tuning is performed on TAPT models and uncased DistilBERT models as listed in Appendix E and F. Models are on default run with 3 epochs. In addition, for each data augmentation method, the TAPT models are run with 6 epochs during the second fine-tuning step to see if increasing the number of epochs improves the performance. The uncased model is also finetuned with 3, 6, and 9 epochs, with 3 epochs being the baseline, to understand the impact of training iterations on non-TAPT models. We apply re-initialization right before the second finetuning process of TAPT models, where the top 1 or 2 blocks are reset. Furthermore, the uncased DistilBERT model is finetuned on in-domain training data with 1,2, and 3 blocks re-initialized to further understand the effect of the hyperparameter on non-TAPT models. All of the other hyperparameters, such as learning rate and batch size, are kept the same as the default values listed in `args.py`.

Lastly, we try ensembling multiple combinations of created models with the Simple and the Weighted Average methods. The combinations that are tested are listed in Appendix D. We then pick the final model that has the highest F1 score.

4.4 Results [1] [2]

Task-Adaptive Pretraining (TAPT) Appendix B shows the results for continued pretraining on the baseline model on out-of-domain training data. TAPT enhances the baseline model's performance only on *relation_extraction*, while worsening it on all other out-of-domain tasks. These results indicate that TAPT on the baseline model does not help it learn more out-of-domain knowledge.

Results for TAPT without the baseline model are shown in Appendix C. Both of the tested TAPT models performed better than the benchmark, which suggests that TAPT was beneficial when applied to DistilBert. The TAPT model pretrained on out-of-domain data with contexts and questions augmented performed the best, so it was chosen as the final pretrained model. Therefore, we believe that augmentation on both contexts and questions also makes the pretraining on limited training data more effective.

Data Augmentation For Second Stage Finetuning Results from Table 1 show that all data augmentation techniques help boost the baseline's performance. In fact, data augmentation is necessary because finetuning directly on the unaugmented out-of-domain decreases the F1 score. Augmentation based on Word2Vec performs the worst, while both WordNet and EDA SR increase F1 score by over 2.1%. One possible reason is that Word2Vec is more likely to return nonsensical augmented sentences.

[1]The naming system used in this project is as follows: ① each training stage is separated by "+" ② the training data used is specified after "_". For example, `Baseline + Finetune_augoodomain (WordNet)` means load the `Baseline` model, and then finettune it on augmented out-of-domain training set with the WordNet augmentation technique.

[2]Test scores are provided in 4.4. All the other scores reported in this section are dev scores.

Models	F1	EM
Baseline	48.43	33.25
Baseline + Finetune_oodomain	48.16	33.25
Baseline + Finetune_augoodomain (Word2Vec)	48.53	33.77
Baseline + Finetune_augoodomain (WordNet)	49.47	35.08
Baseline + Finetune_augoodomain (EDA SR)	49.31	34.29
Baseline + Finetune_augoodomain (back translation)	48.85	33.25

Table 1: Baseline + Data Augmentation Finetuning Experiments Comparison

Models	F1	EM
Pretrain_augoodomain + Finetune_indomain	48.45	32.20
Pretrain_augoodomain + Finetune_indomain + Finetune_oodomain	48.41	32.46
Pretrain_augoodomain + Finetune_indomain + Finetune_augoodomain (Word2Vec)	48.26	32.20
Pretrain_augoodomain + Finetune_indomain + Finetune_augoodomain (WordNet)	49.45	34.29
Pretrain_augoodomain + Finetune_indomain + Finetune_augoodomain (EDA SR)	**51.16**	**36.13**
Pretrain_augoodomain + Finetune_indomain + Finetune_augoodomain (back translation)	49.52	35.34

Table 2: TAPT on augmented oodomain (both contexts and questions augmented) + Data Augmentation Finetuning Experiments Comparison

When using the best TAPT model and repeating the experiments in Table 2, we see that the results show a similar pattern. The last two models have particularly good performances. `Pretrain_augoodomain + Finetune_indomain + Finetune_augoodomain (EDA SR)` has the best validation score of 51.16 F1 score and 36.13 EM score.

Method	Model 1	Model 2	Model 3	F1	EM
Simple	TAPT (WordNet) + reinit 1 + epoch 6	TAPT (back translation)	TAPT (EDA SR)	50.49	35.60
Simple	Baseline	TAPT (back translation)	TAPT (EDA SR)	**52.57**	**37.96**
Weighted Avg	TAPT (WordNet) + reinit 1 + epoch 6	TAPT (back translation)	TAPT (EDA SR)	44.71	30.37
Weighted Avg	Baseline	TAPT (back translation)	TAPT (EDA SR)	39.55	25.39

Table 3: Ensemble Model Results

[1] TAPT: Pretrain_augoodomain + Finetune_indomain + Finetune_augoodomain with no re-init, 3 epochs

Hyperparameter Tuning, Ensembling Method, and Final Model Re-initialization of top transformer blocks hurt performance most of the time on both TAPT models and uncased-DistilBERT-based models, as shown in Appendix E. Training models for more epochs had minimal positive impacts (less than 0.01) on most of the TAPT models. However, when the baseline model is trained for longer (6 epochs), its F1 score increases by 0.77 and its EM score by 0.26. When the number of epochs was increased to 9, the scores remained the same again.

Models	F1	EM
Single Best	58.27	41.49
Ensemble Best	60.42	43.42
Ensemble Second-Best	60.42	43.42

Table 4: Final Candidate Model Performance on Test Set

For creating ensembles, we first combine three models with the highest scores so far, TAPT EDA SR, TAPT back translation, and TAPT WordNet model with 1 block re-initialized. First, we test the Simple method of creating predictions from highest probabilities. To our surprise, this decreases the F1 and EM scores compared to the best single model. However, when the baseline is included in the ensemble, the performance shoots up, producing our final model. The weighted average method consistently produces models with lower performance than the simple method. Other model and ensemble method combinations that are tested and their performances can be found in Appendix D.

Test Leaderboard Results We test the ensemble model with the best dev score, the single model with the best dev score, and the ensemble model with the second-best dev score, on the test set (Baseline + FT_oodomain (Word2Vec), Baseline + FT_oodomain + epoch 9, TAPT (EDA SR)). The final model we choose is the ensemble of baseline + TAPT (back translation) + TAPT (EDA SR) model with test F1 score of 60.42. This model has a high EM score compared to others on the leader board, likely due to the ensembling method selecting the predictions with the highest probabilities. The scores of the final three candidate models on the test set are listed in Table 4.

5 Analysis

Method Analysis In our experiment, we notice that TAPT on baseline worsens the model's performance. This result agrees with the conclusion from the paper[5] that cross-task transfer is sometimes harmful. We conclude that adapting a large but general corpus (in-domain data in this case) does not necessarily improve the model's performance on all tasks.

The TAPT model is pretrained on oodomain contexts and questions, augmented by EDA SR. And we see that EDA SR often performs the best with TAPT. Thus, we conjecture that TAPT models learn better when their pretraining and finetuning stages see data augmented with similar techniques.

Regarding hyperparameters, increasing the number of training epochs shows a positive impact on DistilBert whereas the impact is minimal on TAPT models. We hypothesize that this is because the TAPT models have already been pretrained and have their data augmented to improve their performance on fewshot learning. Because we see score improvements on the uncased model, we believe that training for longer does improve performance even when the model is trained on larger datasets (the uncased models were finetuned on in-domain data). Another pattern observed is that re-initialization rarely helps. For TAPT models, this may be because the data, despite being augmented, is not enough to produce appropriate weights from re-initialized weights.

Prediction Analysis To better understand how our model works and why it fails, we conducted error analysis on the prediction results. First, Figure 2 shows that our best model outperforms the baseline model on DuoRC and RelationExtraction, yet achieves a slightly lower F1 score on RACE. This may be a result of TAPT, which may have hurt RACE performance due to cross domain training.

We then look at the difference between our predictions and the true answers. Figure 3 shows the relationship between F1 score and length difference: predicting a slightly longer answer does not hurt F1 score as much as predicting a shorter answer. We observe that the best model tends to guess an average length rather than adapt to cases where the true answers are very short or very long, which is further confirmed by the third plot. Breaking down the average length of predictions and answers of the three datasets, we can see that the average prediction length (blue) is between 3 and 5 in Figure 3. The values are higher than those of the answer length (orange), considering cases of extremely long answers are rare.

We also conduct detailed analysis on 5 wrong predictions of questions in out-of-domain validation set (see Appendix G). The main reasons why the model fails is that the model uses wrong part of the context to find the answer as this case can be seen in four out of five examples. When the model searches answers by simply looking for exact same words in context and questions instead of looking for lexical variation, the wrong part of context is likely to be used for prediction. Also, when an answer needs multiple sentence reasoning, the model is likely to miss some pieces of evidence and give the wrong answer. Lexical variation can be learned by continued pretraining on related context. The architecture of the model might needs to be changed to improve the model's ability in logical reasoning and multiple sentence reasoning. For example, a model with memory of larger context may have better performance in multiple sentence reasoning.

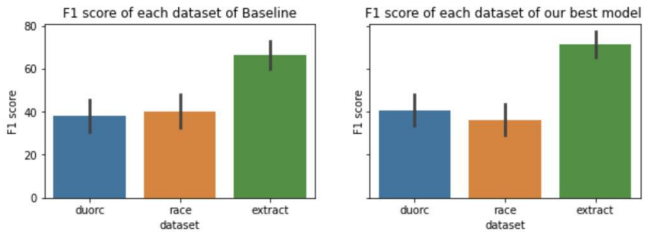


Figure 2: F1 scores across out-of-domain validation sets

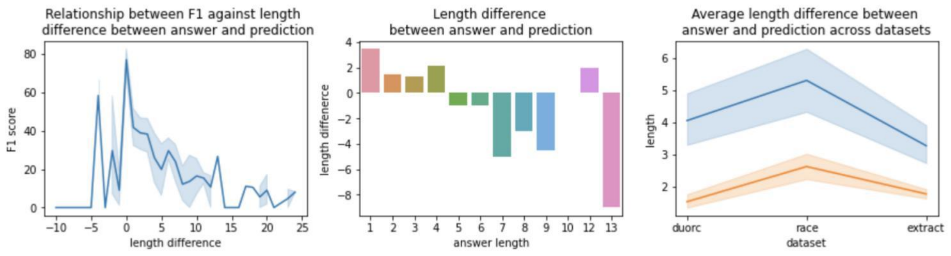


Figure 3: Analysis on length difference between true answer and prediction

① F1 scores against length difference between and answer and prediction ② Length difference between answer and prediction against true answer length on all out-of-domain validation sets ③ Average length of predictions and answers broken down by three validation sets (Blue line indicates prediction length and orange line indicates answer length)

6 Conclusion

In this paper, we investigate whether (1) task-adaptive pretraining (TAPT), (2) data augmentation, and (3) hyperparameter tuning/ensemble methods can improve a model's performance on out-of-domain question answering task. By combining these techniques, we are able to build a model that achieves F1 = 60.42 and EM = 43.42 on the test set.

From our experiments, we learned that TAPT can boost performance when applied to models that have not been finetuned. However, it can be harmful when there is cross-task transfer. Our experiment also shows that data augmentation on training data improves performance. Among the four data augmentation techniques we used, EDA SR is the most effective, while Word2Vec is the least effective. We conclude that data augmentation for limited out-of-domain data is not only beneficial but also necessary when finetuning to adapt to the unseen domain. For future work, we could experiment with the number of words to replace in augmentation. More words replaced might increase the robustness of the QA system but might also introduce noise. For hyperparameter tuning, we show that higher number of training epochs improves performance, but the degree of improvement depends on how much knowledge the model has before training. Our results also show that re-initialization does not help, and the limited amount of training data is the possible reason. Further studies with larger data for finetuning could help confirm our hypothesis about the benefits of re-initialization.

References

[1] Robin Jia and Percy Liang, *Adversarial Examples for Evaluating Reading Comprehension Systems*, arXiv:1707.07328,2017.

[2] Suchin Gururangan and Swabha Swayamdipta and Omer Levy and Roy Schwartz and Samuel R. Bowman and Noah A. Smith, *Annotation Artifacts in Natural Language Inference Data*, arXiv:1803.02324, 2018.

[3] McCoy, Tom and Pavlick, Ellie and Linzen, Tal, *Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference*, Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, July, 2019, 3428–3448.

[4] Marco Tulio Ribeiro and Tongshuang Wu and Carlos Guestrin and Sameer Singh, *Beyond Accuracy: Behavioral Testing of NLP models with CheckList*, arXiv:2005.04118, 2020.

[5] Suchin Gururangan and Ana Marasović and Swabha Swayamdipta and Kyle Lo and Iz Beltagy and Doug Downey and Noah A. Smith, *Don't Stop Pretraining: Adapt Language Models to Domains and Tasks*. arXiv preprint arXiv:2004.10964, 2020.

[6] A. Wang et al. "Superglue: A stickier benchmark for general-purpose language understanding systems". In: Advances in Neural Information Processing Systems. 2019, pp. 3266–3280.

[7] Jason Wei and Kai Zou, *Eda: Easy data augmentation techniques for boosting performance on text classification tasks*. arXiv preprint arXiv:1901.11196, 2019.

[8] Marivate, Vukosi and Sefara, Tshephisho, *Improving Short Text Classification Through Global Augmentation Methods*, Machine Learning and Knowledge Extraction, Springer International Publishing, 385–399, 2020.

[9] Sugiyama, Amane and Yoshinaga, Naok, *Data augmentation using back-translation for context-aware neural machine translation*, Proceedings of the Fourth Workshop on Discourse in Machine Translation (DiscoMT 2019, Association for Computational, 35–44, 2019.

[10] V. Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: arXiv preprint arXiv:1910.01108 (2019).

[11] Thomas Wolf and Lysandre Debut and Victor Sanh and Julien Chaumond and Clement Delangue and Anthony Moi and Pierric Cistac and Tim Rault and Rémi Louf and Morgan Funtowicz and Joe Davison and Sam Shleifer and Patrick von Platen and Clara Ma and Yacine Jernite and Julien Plu and Canwen Xu and Teven Le Scao and Sylvain Gugger and Mariama Drame and Quentin Lhoest and Alexander M. Rush. Transformers: State-of-the-Art Natural Language Processing. Association for Computational Linguistics, 2020.

[12] Shayne Longpre, Yi Lu, Zhucheng Tu, and Chris DuBois. *An exploration of data augmentation and sampling techniques for domain-agnostic question answering.* arXiv preprint arXiv preprint arXiv:1912.02145, 2019.

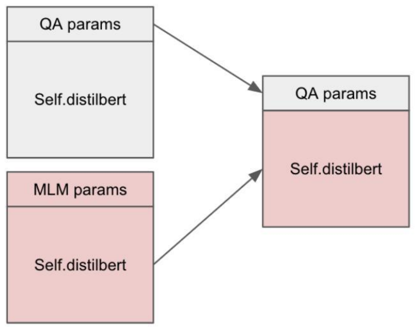
[13] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. CoRR, abs/1606.05250, 2016.

[14] 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.

[15] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bach- man, and Kaheer Suleman. Newsqa: A machine comprehension dataset. ACL 2017, page 191, 2017.

[16] Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, Yoav Artzi. Revisiting Few-sample BERT Fine-tuning. arXiv preprint arXiv:2006.05987, 2020.

A TAPT method in Section 4.1.1: loaded the shared weights from a pretrained MLM model to the baseline QA model.



Grey: from baseline. Pink: from pretrained MLM

B Comparison of models in Section 4.1.1 TAPT with the baseline model

Validation data / Model	all oodomain	RARC	RelationExtraction	DuoRc
Finetune_indomain (Baseline)	F1: 48.43 EM: 33.25	F1: 40.04 EM: 28.12	F1: 66.51 EM: 42.19	F1: 38.59 EM: 29.37
Pretrain_Baseline_oodomain + Finetune_oodomain	F1: 43.03 EM: 30.10	F1: 29.63 EM: 18.75	**F1: 66.78 EM: 43.75**	F1: 32.52 EM: 27.78

C Comparison of models in Section 4.1.2 TAPT without the baseline model

Model	F1	EM
Finetune_augoodomain (back translation) (Benchmark for TAPT)	26.68	17.80
Pretrain_augoodomain (only questions augmented) + Finetune_augoodomain (back translation)	28.51	17.54
Pretrain_augoodomain (both contexts and questions augmented) + Finetune_augoodomain (back translation)	**30.05**	**18.85**

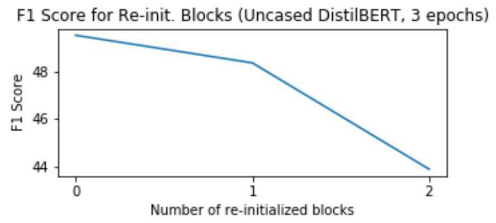
D Ensemble Results Comparison

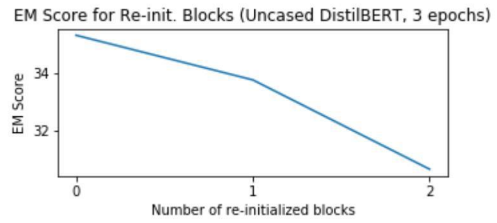
Method	Model 1	Model 2	Model 3	F1	EM
Simple	TAPT (WordNet) + reinit 1 + epoch 6	TAPT (back translation)	TAPT (EDA SR)	50.49	35.60
Simple	Baseline	TAPT (back translation)	TAPT (EDA SR)	**52.57**	**37.96**
Simple	Baseline + FT_oodomain (Word2Vec)	Baseline + FT_oodomain + epoch 9	TAPT (EDA SR)	52.12	36.91
Weighted Avg	TAPT (WordNet) + reinit 1 + epoch 6	TAPT (back translation)	TAPT (EDA SR)	44.71	30.37
Weighted Avg	Baseline	TAPT (back translation)	TAPT (EDA SR)	39.55	25.39
Weighted Avg	Baseline + FT_oodomain (Word2Vec)	Baseline + FT_oodomain + epoch 9	TAPT (EDA SR)	38.46	22.51
Simple	TAPT (WordNet)	TAPT (back translation)	TAPT (EDA SR)	50.93	35.34
Simple	Baseline	Baseline + FT_oodomain (WordNet)	TAPT (EDA SR)	50.30	36.91
Simple	TAPT (back translation)	Baseline + FT_oodomain (WordNet)	TAPT (EDA SR)	51.11	36.91
Simple	Baseline	Uncased + FT_oodomain + epoch 6	TAPT (EDA SR)	47.39	35.34

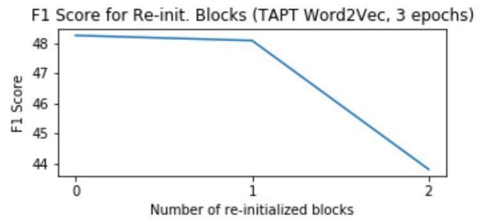
[1] TAPT: Pretrain_augoodomain + Finetune_indomain + Finetune_augoodomain with no re-init, 3 epochs

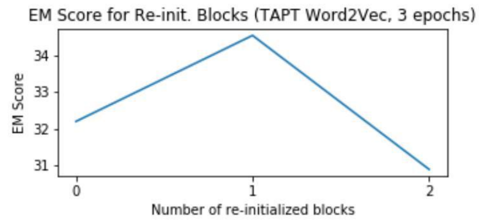
[2] FT_oodomain: Fine-tuned on out-of-domain datasets

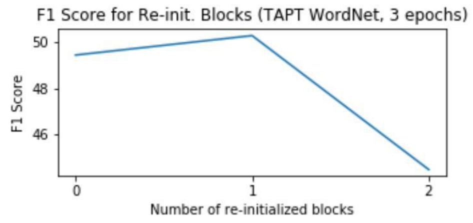
E Hyperparameter Tuning: Reinitialization Performance Plots

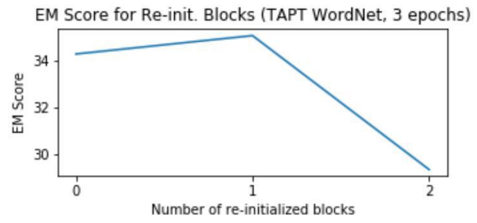


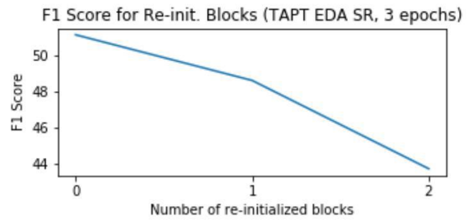


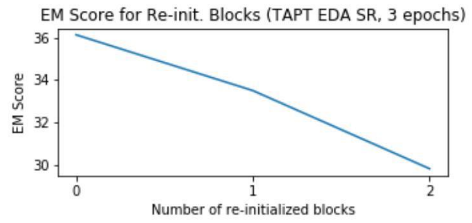


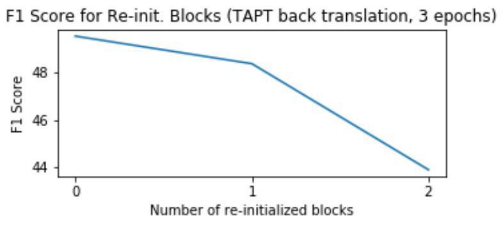


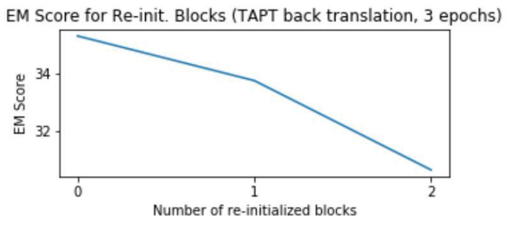












F Hyperparameter Tuning Results

Pre-train	Fine-tune 1	Fine-tune 2	Data Aug. Method	Reinitialization	Epoch	F1	EM
None	In-domain	None	None	0	3*	48.43	33.25
					6	49.20	33.51
					9	49.20	33.51
				1	3	46.89	31.15
				2	3	37.80	22.25
				3	3	33.21	19.63
AugOOD	In-domain	AugOOD	Word2Vec	0	3**	48.26	32.20
					6	48.56	35.34
				1	3	48.09	34.55
					6	48.09	34.55
				2	3	43.81	30.89
					6	44.16	30.89
AugOOD	In-domain	AugOOD	WordNet	0	3**	49.45	34.29
					6	49.45	34.29
				1	3	50.29	35.08
					6	50.29	35.08
				2	3	44.50	29.32
					6	44.60	30.89
AugOOD	In-domain	AugOOD	EDA SR	0	3**	51.16	36.13
					6	51.16	36.13
				1	3	48.63	33.51
					6	48.63	33.51
				2	3	43.74	29.84
					6	43.74	29.84
AugOOD	In-domain	AugOOD	back translation	0	3**	49.52	35.34
					6	49.52	35.34
				1	3	48.36	33.77
					6	48.36	33.77
				2	3	43.90	30.63
					6	44.32	30.63

* Represents baseline
** Also included in Table 2

G Examples of Errors in Predictions of Answerable Questions in out-of-domain Validation Set

Shorten Context	Question	Answer	Prediction	Skill-Deficiency
When they find themselves in a room with trapped rooms all around and below, Quentin checks the door in the ceiling... Leaven's theory that **non-prime-numbered rooms are safe is shown to be incorrect**.	Which rooms does Leaven think are traps?	Prime numbers	all around and below	Lexical variation (anatomy)
He exemplifies how to **test for traps** by tossing a boot into the rooms while holding onto the laces, to trigger potential traps, figuring that the trapped room contains motion detectors.	What do some of the rooms contain?	Traps	Electrochemical sensor	Logical reasoning & Syntactic variation
Soon after, Rennes jumps into a room tested with a boot, and is sprayed in the face with acid. The others pull him back, but **he dies as the acid corrodes his face** and the inside of his head.The group decides that the room must have contained an electrochemical sensor which Rennes missed.	What kind of spray is Rennes killed by?	acid	acid. The others pull him back, but he dies as the acid corrodes his face	Extract key information
Leaven reveals herself to excel at mathematics, and after looking at the numbers on a crawlspace, theorizes that when one of those numbers is prime, the room is booby-trapped. **Leaven's purpose becomes attempting to crack the Cube's code**, and they progress through the cubes.	Who has knowledge of the Cube?	Leaven	Group	Logical reasoning
Vince Rommel (Ross Hagen) is **a good sidehacker runner. Sidehacking is a way of running motorcycle races** with a passenger on the sidecar/sidehack,and Vince is supposed to be one of the best riders....Nero tells Rommel that there's an upcoming sidehack bike race.	Rommel is a mechanic and what kind of racer?	Sidehacker runner	Bike Race	Lexical variation (synonymy) & Multiple sentence reasoning

[*] Words relevant to the corresponding reasoning type are bold, and words relevant to the predictedanswer are underlined