Fine-tuning Strategies for Domain Specific Question Answering under Low Annotation Budget Constraints

Anonymous ACL submission

Abstract

The progress introduced by pre-trained language models and their fine-tuning has resulted in significant improvements in most downstream NLP tasks. The unsupervised fine-tuning of a language model combined with further target task fine-tuning has become the standard QA fine-tuning procedure. In this work, we demonstrate that this strategy is sub-optimal for fine-tuning QA models, especially under a low QA annotation budget, which is a usual setting in practice due to the extractive QA labeling cost. We draw our conclusions by conducting an exhaustive analysis of the performance of the alternatives of the sequential fine-tuning strategy on different QA datasets. Our experiments provide one of the first investigations on how to best fine-tune a QA system under a low budget, and is therefore of the utmost practical interest for the QA practitioner.

1 Introduction

In the recent few years, transformer-based language models like BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), T5 (Raffel et al., 2019)) and GPT-3 (Brown et al., 2020), play a vital role in the Natural Language Processing (NLP) community. Being trained on a tremendous amount of data in a unsupervised fashion, they become the *de facto* starting point of any modern NLP pipeline. The reason is that the adaptability to new tasks of these so-called *foundation models* (Bommasani et al., 2021) has led to substantial improvements in many NLP downstream tasks, such as sequence classification (González-Carvajal and Garrido-Merchán, 2020), text summarization (Miller, 2019), text generation (Raffel et al., 2019) and question answering (Yang et al., 2019). However, this adaptability comes at a cost: adapting foundation models to a specific and complex task requires a significant amount of annotated samples in order to fine-tune those models to the task at hand (Antonello et al., 2021). In practice, the training datasets for domain specific tasks are usually rather small due to budget constraints. Having access to hundred of labeled samples for a task is common and is not tagged as a few-shots scenario, yet the limited annotation budget still makes the fine-tuning task tedious. To circumvent this issue, a double fine-tuning step is usually introduced. It consists of fine-tuning the pre-trained foundation model on a large scale training dataset which is as close as possible (domain and objective) to the target task, and is then further fine-tuned on the given domain/task for which training data is scarce. The result is a model that had been trained as a foundation model, which is a Pre-trained Language Model (`PLM`) like BERT (Devlin et al., 2019), then fine-tuned on a large-scale more specific task (`LM'`), and ultimately refined on the domain/task at hand (`LM''`). Note that this is applied sequentially. In this work, we explore how to best fine-tune models for domain-specific Question Answering (QA) with limited training data. In this paper, we consider extractive QA, and we therefore denote QA as the task of answering questions asked in natural language and finding the answer text-span in a document containing the answer. In the double fine-tuning step stated above, we can use Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) which is a high-quality QA dataset that covers diverse knowledge for the `PLM` to train on. Using a transform-based language model as the `PLM` starting point, PLM fine-tuned on SQuAD can therefore act as the intermediary model (`LM'`) for domain specific question answering. This model is the go-to choice for general QA scenarios. The performance of these state-of-the-art QA models have gained traction and given more attention to the QA task. Nonetheless, in many real-life scenarios, specific-domain QA has a range of field applications which is narrower than SQuAD and may not appear in the SQuAD training data. This calls for building domain-specific dataset to further fine-tune a QA model for the domain at hand to produce a QA model LM″. This last fine-tuning step is domain-dependent, and the practitioner's goal is also to ultimately keep the number of annotated training samples low - he is under a low annotation budget constraint. In practice, such an annotation task pairs remains achievable for a couple of hundred QA examples and for a single domain, but it hardly scales to all the different domains that a company building QA systems needs to deal with – producing questions and annotating them with their corresponding text fragment in a document is much more difficult and time-consuming than creating text classification labels for instance. As a consequence, the assumption is that a limited number of training QA pairs are usually provided to fine-tune LM″ to prepare its domain drift towards the domain specific QA task. In this paper budget refers to the number of domain-specific annotations available at the time of fine-tuning LM″. The main pending issue is how to best fine-tune a QA model down to a target domain under such low annotation budget conditions.

Our contribution is a study of the different strategies one can use to fine-tune a domain-specific extractive QA model. This study is exhaustive as we report experiment for 108 different strategies, applied to 4 different datasets (we discussed 432 trained models, each ran 5 times, see Section 4). We provide a complete protocol and evaluation scheme freely available to the community[1]. Based on these contributions, we explored different low annotation budget scenario for which our findings are as follows: (1) We demonstrate that the standard sequential QA fine-tuning strategy is sub-optimal for QA under a budget, (2) Contrary to reasonable expectations, fine-tuning the text encoder using masked language modeling on domain corpus prior task fine-tuning does not provide an improvement (we even consistently observed a slight degradation of performance), (3) A very low annotation budget goes a long way, that is 200 annotated QA pairs is very efficient with respect to the annotation required, (4) We demonstrate that is it better to go either with a small annotation budget with a careful choice of the fine-tuning strategies, or to go for more than 1,600 annotations. Anything doubling of the annotation budget in between only resulting in a 2% improvement in rare cases.

[1]code and dataset are available in a github repository, private during the review process

2 Related Work

In Question Answering there are mainly three fine-tuning strategies to adapt a language model to a specific domain. These strategies are non-exclusive so that the standard process to create a domain extractive QA system is to apply them as a sequential pipeline as depicted in Figure 1. In what follows, we describe and discuss the related works to each of these fine-tuning steps.

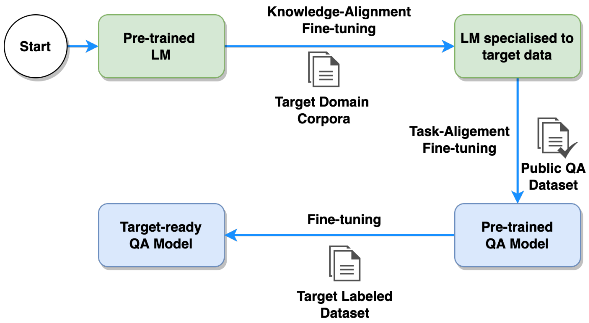

Figure 1: Mainstream methods for QA fine-tuning.

2.1 Knowledge-Alignment Fine-tuning

Knowledge-Alignment Fine-tuning aims to integrate information about the underlying text corpus into the LM. It is often achieved using masked language modeling task, inherited from the LM pre-training objective. It helps aligning the knowledge from the target domain which can be substantially different from what the used LM is pre-trained on. For different NLP tasks this fine-tuning strategy has shown performance improvements. For example (Lee et al., 2019) fine-tunes BERT via knowledge-alignment on Biological corpora (PubMed). The corresponding model BioBERT, can outperforms the BERT model in many biomedical text mining tasks like Named Entity Recognition (NER), Relation Extraction (RE) and QA. Similarly (Nguyen et al., 2020) generate BERTweet by knowledge-alignment fine-tuning with 850M English tweets. The resulting model gets improvements in part-of-speech tagging, NER, and text classification. Nonetheless, it has been shown in (Zhao and Bethard, 2020) that the benefits vary depending on the task and on the flavor (base or large) version of BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) models. (Edwards et al., 2020) also reports difficulties to fine-tune a BERT model with a limited domain corpus, which is usually the case for domain-specific extractive QA.

2.2 Task-Alignment Fine-tuning

Task-Alignment Fine-tuning aims at adapting the pre-trained LM to the target task, that is extractive QA in the scope of this work. Generally, publicly available large datasets are used for this purpose. In the QA domain, the dataset of choice is SQuAD (Rajpurkar et al., 2016) as it contains more than 100k questions on significantly different topics. In order to obtain a neural extractive QA model, the LM is used as a text encoder, then two independent softmax layers are added on top of it in order to predict the start index and stop index of the answer span (Devlin et al., 2019). The added layers being shallow, the LM weights are not frozen, and the training therefore updates the LM parameters with respect to the extractive QA task. This fine-tuning strategy is for example used in (Kratzwald et al., 2020a; Möller et al., 2020). However, in (Merchant et al., 2020), the authors demonstrate that SQuAD-based fine-tuning involves only shallow changes to the LM and mostly to its top layers. In (Cui et al., 2019) the authors try to alleviate this issue by introducing sparse attention in BERT attention heads when fine-tuning – this however comes to a complexity cost and a modest improvement on SQuAD fine-tuning. Few methods take interests in finding significantly different alternatives, but in (Khashabi et al., 2020), the authors do take an opposite stance. They train a unified QA model, where unified means able to perform multiple form of QA (extractive, multiple choice, etc.). As the model is trained to generalize to different QA task formats and still performs well on all domain tasks, it is afterwards fine-tuned on each target dataset which ultimately leads to as many QA models.

2.3 Target Data Fine-tuning

Target Data Fine-tuning is adapting the LM to the target task using target task labeled training data. This allows considerable performance improvements, it is limited by the amount of training data. There are different variations of this strategy. Either the model is directly fine-tuned using the target data (as done in (Kratzwald et al., 2020a; Möller et al., 2020)) or it is trained on a mix between general QA questions and the target ones. In the latter case questions from SQuAD can be used to mix as it is done in (Kratzwald and Feuerriegel, 2019). The authors only explore one way to combine these data, and it is therefore not assumed here that this is the best strategy.

To summarize, the most common fine-tuning strategy used in literature for domain adaptation in QA is as follows: first the pre-trained language model (`PLM`) is optionally further pretrained in a unsupervised fashion on the domain corpus at hand using the masked language modeling task (`PLM+`), second the `PLM` (or optionally `PLM+`) is fine-tuned on SQuAD via Task-Alignment Fine-tuning (`LM'`), and third the network is fine-tuned again on the domain QA pairs annotations that one may have (`LM''`). We are not aware of any study that tried to compare the different fine-tuning strategies and also considered several ways to combine SQuAD and domain-specific corpora when fine-tuning domain-specific extractive QA systems.

3 Methodology

We considered the following options.

MLM Knowledge-Alignment Fine-tuning: In our experiments, we used the Masked Language Modeling (MLM) task to distill the knowledge of the corpora into the LM as discussed in Section 2.1. To assess whether knowledge-alignment fine-tuning via MLM helps improve performance under low-budget situations, we conduct the experiments both with and without this procedure for all combinations of fine-tuning strategies presented in Section 3.1.

SQuAD Fine-tuning: Following explanations from Section 2.2, we include the possible steps to build a pretrained QA model based on SQuAD.

Target Data Fine-tuning and Domain Drift Boosting: Target data fine-tuning (training on the domain labelled QA samples) usually happens after fine-tuning the text encoder on SQuAD, a high-quality rich dataset for aligning the model to the open domain QA setting. However, when it comes to a dataset that is substantially different from SQuAD, both in wording and syntax, this method may become undesirable due to the significant domain drift (Elsahar and Gallé, 2019). Furthermore, it is known that LMs tend to behave unstable (Mou et al., 2016) and lean to overfit the dataset. Since we are experimenting on low budget situations, this effect is amplified and should be avoided. In order to solve this problem, we explore the option to merge the SQuAD and Target QA dataset together in order to make the fine-tuning process stable and avoid the catastrophic forgetting usually happening in QA fine-tuning. The merged fine-tuning approach can benefit from the original hyper-parameters used in SQuAD fine-tuning and bypass the errors that may occur during extensive hyper-parameters searching. Ultimately we would want to have as many target samples than general samples, but accumulating high-quality training datasets of SQuAD's size for every domains is expensive and hardly realistic (Section 1). Inspired by the techniques from classification with imbalance classes, we choose to undersample or oversample the datasets in order to put more emphasis on the domain-specific data. In our case, options available will be either undersampling SQuAD or oversampling the target dataset. The expectation is that the model does not overfit the target training data as it has also to optimize the general QA training samples that are not included in the domain. Nonetheless, the merging of both general and target QA samples is rarely used in the literature, and the ratio on how to best merge the general and target datasets is heavily understudied. For this reason, we devised different merging options that we will later compare – we will show that merging is actually the best strategy and that all of the merging options are not equal.

To best describe these merging options, we will use the following notations. Let $\mathcal{D}_g$ be the general QA training dataset, $\mathcal{D}_t$ the target QA training dataset and $\mathcal{D}_f$ the final merge training dataset we want to build given a dataset merging option. We then define the following merging options :

- **TargetQA**, that is only the target samples – in other words no merge, s.t. $\mathcal{D}_f = \mathcal{D}_t$
- **MP**, Merge Partial SQuAD based on a a 1:1 merge. Since $|\mathcal{D}_g| >> |\mathcal{D}_t|$, we take all samples from $\mathcal{D}_t$, and we sample n samples from $\mathcal{D}_g$, s.t. $\mathcal{D}_f = \mathcal{D}_t \cup \{s_1, \ldots, s_n\}$ where $s_i \overset{\text{i.i.d.}}{\sim} U(\mathcal{D}_g)$ and $U(\mathcal{D}_g)$ denotes the uniform distribution over the set $\mathcal{D}_g$ and $n = |\mathcal{D}_t|$.
- **MPO**, Merge Partial SQuAD with Oversampling is close to the previous **MP** strategy, but in **MPO** we sample three times the set $\mathcal{D}_t$ so that the resulting merged QA training dataset $\mathcal{D}_f$ is twice larger – in **MPO** the model will see three times more the target samples (as the best reported value in the work (Kratzwald and Feuerriegel, 2019)) in order to amplify the learning signals for the target domain while still having to satisfy the samples sampled from $\mathcal{D}_g$. More formally in that merging option: $\mathcal{D}_f = \mathcal{D}_t \cup \mathcal{D}_t \cup \mathcal{D}_t \cup \{s_1, \ldots, s_n\}$ where $s_i \overset{\text{i.i.d.}}{\sim} U(\mathcal{D}_g)$.
- **MW**, Merge Whole SQuAD, that is the union of both training dataset s.t. $\mathcal{D}_f = \mathcal{D}_t \cup \mathcal{D}_g$. For that merging option, the QA model will be trained on much more training samples for better QA in general, at the expense to learn from a weaker signal coming from the target task. It is interesting to note that under this merging option, the training data are absolutely the same than the mainstream sequential approach to fine-tuning, although there are not drawn sequentially when training but mixed in a single training step. This single difference, embarrassingly simple, accounts however for 5 and up to 10 macro-f1 increase for all datasets but one when the budget is set to 100 annotations.
- **MWO**, Merge Whole SQuAD with Oversampling is close to the previous **MW** strategy but we do the same oversampling as **MPO**, then we have $D_f = D_t \cup D_t \cup D_t \cup D_g$ where $\mathcal{D}_g$ keeps its original size.

We also considered a curriculum learning approach (Bengio et al., 2009), in which more simple QA pairs will be used and we would introduce more and more difficult QA samples as the training progresses. Since evaluating the QA pair difficulty is not trivial, we explore this possibility by brute-force as we generated a large number of experiments with different QA pairs splits that are introduced as the training progress. We observed no significant changes in the target model performances, suggesting that either a curriculum approach is not applicable here, or that there is only a very limited number of QA pair sequences that can actually serve a curriculum learning approach. While this was not the primary focus of our work, the existence or nonexistence of such "golden sequences" has yet to be investigated. Moreover, note that we propose merging options in this paper while, as stated in Section 2, sequential transfer learning (PLM $\rightarrow$ SQuAD $\rightarrow$ TargetQA) is the go-to method used in most, if not close to all, QA model fine-tuning pipeline in practice.

3.1 Fine-tuning combinations

As stated above, there are a series of options that we can choose to improve the performance of QA models fine-tuning. Combining the options in different manner lead to as many fine-tuning strategies. In our experiments, we include strategies that can reasonably yield fine-tuning improvements – we especially discard the combination that first perform fine-tuning on the target dataset and then on SQuAD. The meaningful strategies of fine-tuning options we consider for extractive QA fine-tuning in this work are the following:

- PLM $\rightarrow$ SQuAD
- PLM $\rightarrow$ TargetQA
- PLM $\rightarrow$ SQuAD $\rightarrow$ TargetQA
- PLM $\rightarrow$ SQuAD $\rightarrow$ MP
- PLM $\rightarrow$ SQuAD $\rightarrow$ MPO
- PLM $\rightarrow$ MP
- PLM $\rightarrow$ MPO
- PLM $\rightarrow$ MW
- PLM $\rightarrow$ MWO

All the methods listed above will be experimented with knowledge-alignment fine-tuning (unsupervised masked language modeling on the target document corpus) as well, so that we end up with 18 different fine-tuning strategies.

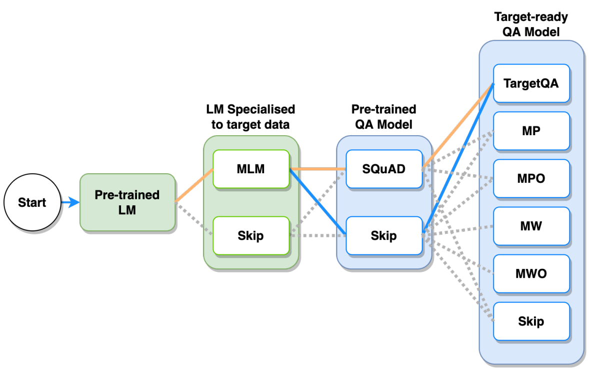

Figure 2: Fine-tuning strategies and combinations considered in our study.

3.2 Datasets

SQuAD is a QA dataset introduced in (Rajpurkar et al., 2016). The dataset contains 100,000 triplets (passages, question, answer). The passages come from 536 Wikipedia articles. The questions and answers are constructed mainly by crowdsourcing: annotators are allowed to ask up to 5 questions on an article, and need to mark the correct answers in the corresponding passage. The major difference between SQuAD and previous QA datasets such as CNN/DM (Hermann et al., 2015), CBT (Hill et al., 2016), etc, is that the answers in SQuAD are not a single entity or word, but may be a phrase, which makes its answers more difficult to predict.

As target domain QA datasets, we consider the following four domain-specific datasets:

COVID-QA (Möller et al., 2020) is a question answering dataset on COVID-19 publications. The dataset contains 147 scientific articles. The quality of the dataset is assured as all the question-answer pairs are annotated by 15 experts with at least a master degree in biomedicine.

CUAD-QA (Hendrycks et al., 2021) contains questions about legal contracts in the commercial domain. The corpus, curated and maintained by the Atticus Project, contains more than $13,000$ annotations in 510 contracts. The original task is to highlight important parts of a contract that are necessary for human to review. We convert it into a question-answering task in SQuAD fashion. The passages to select are lengthy compared to SQuAD paragraphs.

MOVIE-QA contains questions about movie plots extracted from Wikipedia. We constructed the dataset from the DuoRC (Saha et al., 2018) dataset. The original dataset is an English language dataset of questions and answers collected from crowdsourced AMT workers on Wikipedia and IMDb movie plots. It contains two sub-datasets SelfRC and ParaphraseRC. We sampled questions from the SelfRC since the answers of the ParaphraseRC subset are paraphrased from the movies plot.

KG-QA is a dataset that we constructed from the Wikidata knowledge base. It contains keyword questions that are constructed semi-automatically as it is done in a knowledge extraction task using QA techniques borrowed from (Kratzwald et al., 2020b). More specifically, We extracted 982 entities accompanied by their related Wikipedia pages containing predicates like game platform, developer, game mode and etc.

Those four datasets were chosen so that they represent different domains and contain question/answer/context with different characteristics. For the purpose of budget analysis, we randomly sampled $2,000$ examples from each dataset for comparison, and we split our datasets in 5-fold cross validation manner to reduce randomness in our experiments. All datasets are in SQuADv1.1 version i.e. all the questions are answerable.

3.3 Budget Setting

Inspired by the training size analysis in (Edwards et al., 2020), we choose the following experiment budget sizes: 100, 200, 400, 800, 1200, 1600. Those examples are randomly extracted from the training set. As for evaluation of the QA systems in different situations, we use the hold-out test sets (400 examples) for comparison.

3.4 Dataset Analyses

In the following, we are trying to measure the gap between SQuAD and each dataset from different perspectives.

3.4.1 Corpus Analysis

	COVID-QA	CUAD-QA	MOVIE-QA	KG-QA
SQuAD	36.0	34.8	41.4	50.6

Table 1: Vocabulary overlap (%) between domain specific datasets and general dataset SQuAD.

Domain Similarity. We compute a domain similarity metric to objectively identify if a dataset is close or far from SQuAD. We consider the top-10K most frequent unigrams (stop-words excluded) in each datasets and compute the vocabulary overlap (see Table 1). We observed that MOVIE-QA and KG-QA have a stronger similarity with SQuAD dataset than the others. This is reasonable since MOVIE-QA and KG-QA are based on movie plots from wikipedia and wikipedia pages of video game entities respectively. COVID-QA and CUAD-QA are relatively far from SQuAD since the two domains are very specialized either in biology or legal terms.

Dataset	Avg tokens per question	Avg tokens per answer	Avg tokens per document	Corpus size
RoBERTa-PC	-	-	-	160Gb
SQuAD	10.06	3.16	116.64	13Mb
COVID-QA	9.43	13.93	4021.83	50Mb
CUAD-QA	18.53	41.87	8428.79	153Mb
MOVIE-QA	7.35	2.5	601.81	6.8Mb
KG-QA	3.32	1.65	1373.65	17Mb

Table 2: Characteristics of QA datasets used in our experiments. RoBERTa-PC is RoBERTa Pre-training Corpora (PC) reported here for comparison. Candidates are answer candidates in the corpus.

Corpus size: The size of the text corpus is shown in Table 2. Note that the corpus size is a fraction of the corpus used for PLMs.

3.4.2 Question/Answer/Passage Analysis

Different lengths of questions, answers and passages can lead to different inference difficulties. Therefore, the length distribution (see in Table 2) can be a very important metric for evaluating the characteristics of four datasets. First of all, from the answer length, SQuAD together with KG-QA and MOVIE-QA are dominated by short answers. More specifically, the questions in MOVIE-QA are mostly based on character names or dates of specific events. As for KG-QA, the questions are keyword queries and the answers are constructed with the object entities. With this respect, KG-QA and MOVIE-QA can be considered as SQuAD-like but KG-QA is relatively difficult due to the keyword queries. Besides, all the three datasets are based on wikipedia articles with different granularity: SQuAD is the built on top of paragraphs, MOVIE-QA is curated from the movie plots while KG-QA is based on the entire article which can be a bit lengthy. Second, the passages of CUAD-QA are collected from commercial legal contracts in a specific format. It is substantially different than Wikipedia articles in the way of sentence expression and wording choices. Also there is an important gap between CUAD-QA and other datasets in answer length, which infers it is a more difficult and absolutely not SQuAD-like dataset. For COVID-QA, the dataset built on top of biological papers, which is also lengthy to infer. But for the questions and answers, either in the way of asking questions or the type of the answers, COVID-QA is not far from SQuAD. For that matter, COVID-QA is a dataset similar to SQuAD, but relatively more difficult for inference.

Overall one can say that the gap between COVID-QA and MOVIE-QA with SQuAD is smaller than the two other datasets since the questions and answers length as well as the domain are relatively similar.

4 Experiment

4.1 Language Model and fine-tuning strategies

In our experiments we use RoBERTa (Liu et al., 2019) as our starting PLM. RoBERTa is built on BERT: it mainly optimizes key hyper-parameters and simplifies the training objective and training mini-batches size. RoBERTa achieves better performance than BERT in many benchmarks like GLUE (Wang et al., 2019), SQuAD (Rajpurkar et al., 2016) and RACE (Lai et al., 2017), which explains this choice over BERT base or large models. To build extractive QA models, we apply two independent softmax layers for predicting the starting and the ending index of the answer span as in (Devlin et al., 2019). Parameters of softmax layers and the PLM are updated when fine-tuning. As for implementation details, we use the pre-trained, 12-layer, 768-hidden, 12-heads, 125M parameters, RoBERTa base model from Huggingface hub. AdamW (Loshchilov and Hutter, 2019) is used as the optimizer for fine-tuning with learning rate set to $3e-5$. The results are reported using 5-fold cross validation. We explore 18 different fine-tuning combinations (see Section 3.1) for 6 different annotation budget sizes (see Section 3.3) over 4 datasets. Moreover each unique experiment is actually run 5 times for different data splits to get significant results. We therefore hereby provide results based on $2,160$ evaluation runs (with 1918 fine-tuned models). All $2,160$ evaluation runs require 62.5 days of 4 Titan XP GPUs to complete.

4.2 Results

In what follows, we present the main results and analysis we can deduce from our experiments. It is important to note that we provide a summary table with all our results in appendix[2]. We hereby discuss our findings step by step, sometimes with a subset of budget sizes, and the interested reader can analyse the complete experiments table in supplementary materials that compile all the $2,160$ evaluation runs.

(1) The standard fine-tuning strategy for QA is sub-optimal with low training budgets, and although low training budgets are the *de facto* situations in practice (Section 1 in appendix). Out of 24 dataset and budget combinations, it only achieves twice the best performance by a small margin (Table in appendix). On the contrary, the performance difference between the mainstream method and the best fine-tuning strategy identified is up to 12.5% for KG-QA dataset and budget set to 100. The gap is particularly high for low budget ($k = 100$) and tends to be smaller for higher budgets ($k > 800$) see in Table 3. For very low budget ($k = 100$) the average difference is 6.93%, which is very substantial. A reminder here is that such improvement comes at no additional cost for QA practitioners.

[2]Note that this is provided in appendix at reviewing time, and that this page will be included in the paper as camera-ready versions of accepted long papers will be given one additional page of content (up to 9 pages)

	Annotation Budget					
	100	**200**	**400**	**800**	**1200**	**1600**
Baseline	56.70	61.35	64.88	67.45	68.90	69.90
Best strategy	63.63	65.85	67.78	69.83	70.75	71.40
Difference	+6.93	+4.5	+2.9	+2.38	+1.85	+1.5

Table 3: Comparison between the best fine-tuning strategy and baseline strategy: average performance(%) of QA system in different domain (legal, biology, movie plots and video games).

(2) Knowledge-Alignment Fine-tuning has limited improvements in domain-specific QA under a budget. For most of the experiments, we cannot observe that knowledge-alignment fine-tuning (more specifically MLM) steadily and repeatedly improves the accuracy of the models, overall we even consistently observed a slight degradation of performance (Table 4). Moreover, over the few occurrences where MLM helps, it does only by a small margin (Table in appendix). While knowledge-alignment fine-tuning was reported to be helpful for other NLP tasks, our experiments show that this is not the case for low annotation budget extractive QA. We associate this to the corpora size of the domain datasets that are several order of magnitude smaller then the corpora used in other works where MLM was identified to be useful. Large text corpora are rather exceptional in domain specific QA scenarios, we conclude that MLM fine-tuning is generally not advisable.

	Dataset			
	COVID-QA	**CUAD-QA**	**MOVIE-QA**	**KG-QA**
No MLM	55.44	38.70	78.06	77.62
With MLM	52.97	38.97	77,62	63.9
Difference	-2.47	+0.27	-0.44	-0.95

Table 4: Average performance (%) difference after MLM procedure evaluated over all the budgets and strategies.

(3) A low annotation budget goes a long way. Domain-specific training data is assumed to be the best signal to optimize the network in order to achieve better performances. We show here that, fortunately, even a small number of samples lead to significant improvements. To illustrate this, we compare the baseline QA system

(RoBERTaBase-SQuAD) with other fine-tuning strategies. In very low-budget scenarios ($k = 100$) we observe average performance improvements range between 3.4% and $27,8\%$ (Table 5). For high budgets it ranges between 5.2% and 40.8%. This result is partially consistent with the claim by (Hazen et al., 2019) that low budget fine-tuning is actually overestimating the budget in the practical settings since we have just shown that it depends on the domain drift from SQuAD.

	Dataset			
	COVID-QA	**CUAD-QA**	**MOVIE-QA**	**KG-QA**
Zero-shot	53.8	12.2	80.2	41.9
Low Budget	62.3	40.0	83.6	68.6
Difference	+8.5	+27.8	+3.4	+26.7
High Budget	67.3	50.2	85.4	82.7
Difference	+13.5	+38.0	+5.2	+40.8

Table 5: Performance difference (%) between Zero-Shot scenarios and Few-Shot scenarios with low budget ($K = 100$) and high budget sizes $K = 1,600$

(4) Do not compromise: either go small or big annotation budget. One of the main issue for the practitioners is to know what improvement to expect if one invests more in the QA pair annotation budget - we remind here that building such annotations are more difficult and therefore costly than text classification for instance. We compare the performance improvements that one can achieve by increasing the number of training data – more training data obviously tend to lead to better models, but we want to measure here how worthy is increasing training dataset size. For instance, what are the expectations a practitioner can have if he is willing to double his annotation effort? To answer this question, we compare the best performing fine-tuning strategy for each budget (between $K = 100$ and $k = 1,600$) for the different datasets, assuming that a practitioner is also able to run all strategies to compare them and pick the best one for each budget. We observe the relative gain for each budget jump as reported in Figure 3.

From this experiment we conclude the following. First, providing a small annotation budget (100 or 200) samples is very efficient with respect to a zero shot setting (as discussed in the previous experiment). But we also note that doubling the annotation effort lead to only a 1% performance improvement in general and 2% at a maximum. In practice, doubling the amount of extractive QA labels available for target domain fine-tuning is very expensive and therefore do not justify the average 1% improvement (it is also supposed that the experiments were run for all strategies and that the best one was selected, which add to the complexity to benefit fully from these 1 up to 2% improvement). Complementary, after investing around 10 times the initial budget, the benefit has accumulated and becomes significant with respect to the effort put into the annotation budget. As a rule of thumb, we would advise to either opt for a 200 annotation budget with a careful selection of the **MWO** fine-tuning strategy, or to invest for an annotation budget $\geq 1,600$ without the need to explore different fine-tuning strategies in this case. Any effort within the $[200; 1,600]$ range imply a weak return with respect to the time and effort to double each time the number of domain annotations.

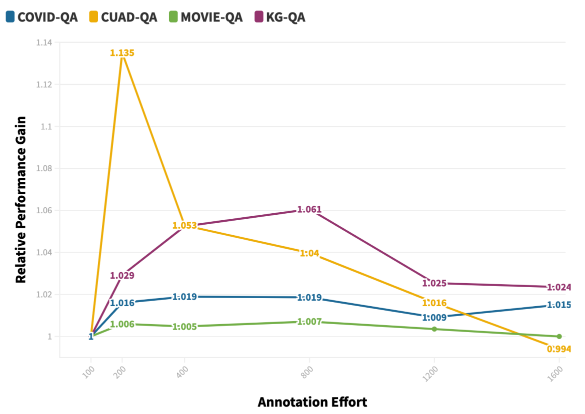

Figure 3: Performance difference (%) after x16 data collection procedure evaluated over low budget ($K = 100$) and high budget sizes ($K = 1,600$).

5 Conclusion

In this work we compared different fine-tuning strategies for extractive QA in low budget scenarios. Our experiments show that the standard fine-tuning strategy for QA is sub-optimal, merge fine-tuning is the most robust and effective fine-tuning strategy, and Knowledge-Alignment Fine-tuning via MLM does not yield a significant improvement. Those are all counter-intuitive results with respect to common practices by the NLP practitioners who usually apply the standard sequential fine-tuning pipeline. We remind that these improvement come at no overhead cost. Finally, our experiments show what are the performance gains that one can expect by collecting different amounts of training data for different domain-specific QA scenarios depending on similarity with SQuAD.

References

Richard Antonello, Nicole Beckage, Javier Turek, and Alexander Huth. 2021. Selecting informative contexts improves language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1072–1085, Online. Association for Computational Linguistics.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 41–48, New York, NY, USA. Association for Computing Machinery.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Baiyun Cui, Yingming Li, Ming Chen, and Zhongfei Zhang. 2019. Fine-tune BERT with sparse self-attention mechanism. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3548–3553, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Aleksandra Edwards, Jose Camacho-Collados, Hélène De Ribaupierre, and Alun Preece. 2020. Go simple and pre-train on domain-specific corpora: On the role of training data for text classification. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5522–5529, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Hady Elsahar and Matthias Gallé. 2019. To annotate or not? predicting performance drop under domain shift. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2163–2173, Hong Kong, China. Association for Computational Linguistics.

Santiago González-Carvajal and Eduardo C Garrido-Merchán. 2020. Comparing bert against traditional machine learning text classification. *arXiv preprint arXiv:2005.13012*.

Timothy J Hazen, Shehzaad Dhuliawala, and Daniel Boies. 2019. Towards domain adaptation from limited data for question answering using deep neural networks. *arXiv preprint arXiv:1911.02655*.

Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. 2021. Cuad: An expert-annotated nlp dataset for legal contract review.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. Unifiedqa: Crossing format boundaries with a single qa system.

Bernhard Kratzwald and Stefan Feuerriegel. 2019. Putting question-answering systems into practice: Transfer learning for efficient domain customization. *ACM Transactions on Management Information Systems*, 9(4):15:1–15:20.

Bernhard Kratzwald, Guo Kunpeng, Stefan Feuerriegel, and Dennis Diefenbach. 2020a. IntKB: A verifiable interactive framework for knowledge base completion. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 5591–5603, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Bernhard Kratzwald, Guo Kunpeng, Stefan Feuerriegel, and Dennis Diefenbach. 2020b. Intkb: A verifiable interactive framework for knowledge base completion. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 5591–5603. International Committee on Computational Linguistics.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*.

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

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

Amil Merchant, Elahe Rahimtoroghi, Ellie Pavlick, and Ian Tenney. 2020. What happens to BERT embeddings during fine-tuning? In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 33–44, Online. Association for Computational Linguistics.

Derek Miller. 2019. Leveraging bert for extractive text summarization on lectures. *arXiv preprint arXiv:1906.04165*.

Timo Möller, Anthony Reina, Raghavan Jayakumar, and Malte Pietsch. 2020. COVID-QA: A question answering dataset for COVID-19. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, Online. Association for Computational Linguistics.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in nlp applications?

Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. 2020. Bertweet: A pre-trained language model for english tweets.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Amrita Saha, Rahul Aralikatte, Mitesh M. Khapra, and Karthik Sankaranarayanan. 2018. Duorc: Towards complex language understanding with paraphrased reading comprehension.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718*.

Yiyun Zhao and Steven Bethard. 2020. How does BERT's attention change when you fine-tune? an analysis methodology and a case study in negation scope. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4729–4747, Online. Association for Computational Linguistics.

A Appendix

Dataset	Fine-tune Strategy	MACRO-F1					
		K = 100	K = 200	K = 400	K = 800	K = 1200	K = 1600
	RoBERTaBase-SQuAD	53.8	53.8	53.8	53.8	53.8	53.8
	RoBERTaBase-MLM-SQuAD	52.9	52.9	52.9	52.9	52.9	52.9
	RoBERTaBase-TargetQA	6.5	35.8	46.6	54.3	55.6	59.2
	RoBERTaBase-MLM-TargetQA	5.6	17.7	35.1	46.8	53.0	54.5
	RoBERTaBase-SQuAD-TargetQA	55.8	58.3	61.0	63.1	64.3	64.1
	RoBERTaBase-MLM-SQuAD-TargetQA	55.0	59.4	60.3	64.2	64.9	64.7
	RoBERTaBase-MP	8.9	44.7	51.7	57.3	59.1	59.9
COVID-QA	RoBERTaBase-MLM-MP	13.9	34.1	45.9	52.6	55.1	59.2
	RoBERTaBase-MPO	27.7	39.4	45.1	50.5	54.3	54.2
	RoBERTaBase-MLM-MPO	18.8	29.3	37.1	47.4	51.2	52.4
	RoBERTaBase-SQuAD-MP	57.1	59.9	62.1	64.0	64.5	64.1
	RoBERTaBase-MLM-SQuAD-MP	56.2	58.0	60.7	63.0	62.6	63.4
	RoBERTaBase-SQuAD-MPO	54.6	58.5	58.0	60.5	60.4	60.8
	RoBERTaBase-MLM-SQuAD-MPO	53.9	56.0	57.0	59.3	58.2	60.2
	RoBERTaBase-MW	60.8	62.6	64.2	**65.7**	**66.3**	**67.3**
	RoBERTaBase-MLM-MW	60.1	63.0	63.2	64.6	65.5	65.9
	RoBERTaBase-MWO	**62.3**	**63.3**	**64.5**	64.1	63.6	64.5
	RoBERTaBase-MLM-MWO	**62.3**	61.2	62.3	62.1	62.8	63.2
	RoBERTaBase-SQuAD	12.2	12.2	12.2	12.2	12.2	12.2
	RoBERTaBase-MLM-SQuAD	14.2	14.2	14.2	14.2	14.2	14.2
	RoBERTaBase-TargetQA	12.2	13.0	39.0	45.9	47.0	48.5
	RoBERTaBase-MLM-TargetQA	12.7	26.2	36.9	44.3	48.0	48.3
	RoBERTaBase-SQuAD-TargetQA	35.6	42.8	45.9	47.1	48.6	**50.2**
	RoBERTaBase-MLM-SQuAD-TargetQA	39.3	44.0	**47.8**	48.2	48.9	49.0
	RoBERTaBase-MP	12.4	34.6	43.4	45.6	48.2	49.5
CUAD-QA	RoBERTaBase-MLM-MP	18.2	31.5	40.1	46.5	47.5	49.1
	RoBERTaBase-MPO	21.4	35.4	40.5	44.8	45.2	45.5
	RoBERTaBase-MLM-MPO	20.0	30.5	35.0	44.2	45.0	45.5
	RoBERTaBase-SQuAD-MP	38.3	42.8	46.1	49.1	49.1	48.7
	RoBERTaBase-MLM-SQuAD-MP	38.0	**45.4**	47.2	**49.7**	49.7	49.8
	RoBERTaBase-SQuAD-MPO	35.6	40.6	43.4	45.0	45.6	45.7
	RoBERTaBase-MLM-SQuAD-MPO	35.8	41.6	44.4	45.4	46.1	46.8
	RoBERTaBase-MW	35.0	42.0	45.6	48.4	**50.5**	50.0
	RoBERTaBase-MLM-MW	34.5	41.5	44.6	48.7	49.4	**50.2**
	RoBERTaBase-MWO	**40.0**	45.0	46.0	46.6	47.8	47.6
	RoBERTaBase-MLM-MWO	39.1	42.7	43.3	45.8	46.2	46.6
	RoBERTaBase-SQuAD	80.2	80.2	80.2	80.2	80.2	80.2
	RoBERTaBase-MLM-SQuAD	80.0	80.0	80.0	80.0	80.0	80.0
	RoBERTaBase-TargetQA	25.0	51.8	67.5	75.0	78.5	80.1
	RoBERTaBase-MLM-TargetQA.	25.9	44.5	54.6	74.9	77.7	79.6
	RoBERTaBase-SQuAD-TargetQA	79.3	79.9	81.9	83.2	83.4	84.0
	RoBERTaBase-MLM-SQuAD-TargetQA	79.7	79.9	82.0	83.5	83.8	83.9
	RoBERTaBase-MP	54.6	61.4	73.3	79.5	80.5	81.8
MOVIE-QA	RoBERTaBase-MLM-MP	52.4	63.8	73.2	78.7	79.7	81.8
	RoBERTaBase-MPO	57.7	66.6	74.2	77.9	80.0	80.7
	RoBERTaBase-MLM-MPO	58.9	67.9	73.3	77.7	79.8	80.2
	RoBERTaBase-SQuAD-MP	79.2	80.4	82.2	83.5	83.6	84.6
	RoBERTaBase-MLM-SQuAD-MP	78.5	80.9	81.0	83.0	84.0	83.3
	RoBERTaBase-SQuAD-MPO	79.7	81.3	82.4	83.3	83.2	83.4
	RoBERTaBase-MLM-SQuAD-MPO	79.4	80.5	81.7	83.6	83.4	82.9
	RoBERTaBase-MW	**83.6**	83.1	**84.5**	84.4	85.1	85.0
	RoBERTaBase-MLM-MW	83.0	82.9	**84.5**	**85.1**	**85.4**	**85.4**
	RoBERTaBase-MWO	82.7	84.0	83.9	84.3	84.8	84.0
	RoBERTaBase-MLM-MWO	83.1	**84.1**	84.3	84.9	84.5	84.5
	RoBERTaBase-SQuAD	41.9	41.9	41.9	41.9	41.9	41.9
	RoBERTaBase-MLM-SQuAD	35.9	35.9	35.9	35.9	35.9	35.9
	RoBERTaBase-TargetQA	20.1	26.2	30.4	70.2	76.1	78.6
	RoBERTaBase-MLM-TargetQA	24.3	27.2	33.6	53.1	73.4	79.1
	RoBERTaBase-SQuAD-TargetQA	56.1	64.4	70.7	76.4	79.3	81.3
	RoBERTaBase-MLM-SQuAD-TargetQA	61.2	66.6	72.6	77.0	79.6	81.5
	RoBERTaBase-MP	24.5	27.0	64.5	76.0	77.9	78.8
KG-QA	RoBERTaBase-MLM-MP	24.3	28.2	43.8	75.2	78.2	80.5
	RoBERTaBase-MPO	28.9	52.2	71.4	76.2	79.9	82.2
	RoBERTaBase-MLM-MPO	40.2	40.2	70.4	77.8	79.9	82.5
	RoBERTaBase-SQuAD-MP	64.6	65.1	73.3	77.1	78.7	81.1
	RoBERTaBase-MLM-SQuAD-MP	65.0	66.7	73.5	77.4	79.0	80.5
	RoBERTaBase-SQuAD-MPO	63.9	69.1	73.5	78.4	**80.8**	82.1
	RoBERTaBase-MLM-SQuAD-MPO	63.9	67.8	73.5	77.5	79.8	81.7
	RoBERTaBase-MW	66.1	68.2	72.3	75.8	77.7	80.4
	RoBERTaBase-MLM-MW	66.2	69.2	73.5	75.8	77.8	81.0
	RoBERTaBase-MWO	67.7	69.3	74.2	**78.8**	80.6	82.5
	RoBERTaBase-MLM-MWO	**68.6**	**70.6**	**74.3**	78.0	**80.8**	**82.7**

Table 6: Experiment results. K is the budget size. *RoBERTaBase-SQuAD-TargetQA* is the standard sequential fine-tuning method, its results are underlined for reference. RoBERTaBase-SQuAD, often referred as the "baseline method" in many benchmarks, reflects how well a SQuAD model generalizes on other QA tasks. Best result for each budget size is given in **bold**.