Improving Data Augmentation in Low-resource Question Answering with Active Learning in Multiple Stages

Anonymous ACL submission

Abstract

Neural approaches have become very popular in the domain of Question Answering, however they require a large amount of annotated data. Furthermore, they often yield very good performance but only in the domain they were trained on. In this work we propose a novel approach that combines data augmentation via question-answer generation and active learning to improve performance in low resource settings, where the target domain is vastly different from the source domain. Furthermore, we investigate data augmentation via generation for question answering in three different low-resource settings relevant in practice and how this can be improved: 1) No labels for the target domain, 2) static, labelled data for the target domain and 3) an Active Learning approach with labels for the target domain provided by an expert. In all settings we assume sufficient amount of labelled data from the source domain is available. We perform extensive experiments in each of the above conditions. Our findings show that our novel approach, which combines data augmentation with active learning, boosts performances in the low-resource, domain-specific setting, allowing for low-labelling-effort question answering systems in new, specialized domains. They further demonstrate how to best utilize data augmentation to boost performance in these settings.

1 Introduction

Machine Reading Question Answering (MRQA) is a challenging and important problem as it facilitates targeted information extraction from documents and allows users to get fast, easy access to a vast amount of documents available, e.g. finding solutions to software errors, or searching for common sense knowledge. As MRQA models need plenty of annotations, several methods exist to augment data by generating new question-answer pairs with the ultimate goal of improving quality of predictions. Some of these approaches show a real benefit in the downstream MRQA task; however, there is no work focusing on employing this kind of data augmentation in low-resource, domain-specific settings often observed in practice. In these cases, most of the times only few annotated samples are available due to the specialized domain being usually vastly different from the source domain of the publicly available labelled data; moreover, labeling is expensive as it requires a significant amount of time from domain experts. However, we argue that it is feasible and not too expensive to collect and annotate a small set of samples (e.g. 200), accurately selected to boost performance of the MRQA model. In this work we introduce a novel approach that follows this idea. Although Shakeri et al. (2020) show that a question-answer generation approach trained on SQuAD (Rajpurkar et al., 2016) transfers fairly well to other domains, this does not hold in the case of datasets from specialized domains that are very relevant in practice. Additionally, so far it stays unclear how few samples from the target domain, selected based on various criteria, affect the performance of the MRQA model when they are used in the training of the sample generation model as well. In this work we introduce a novel approach that enables this and also investigate the performance of data generation for MRQA in different low-resource, domain-specific settings. For the first setting (S1) we assume to have only unlabelled target domain data available in addition to samples from a potentially large source domain and study the generalization to low-resource domains. In addition to that we also consider two other cases with labelled samples from the target domain. For these low-resource settings we distinguish between having static, labelled data available (S2) and having dynamically labelled data (S3), e.g. a domain expert labelling on demand. We employ a state-of-the-art model for generating question-answer pairs on documents from the target domain and test its performance also in S2. For S3, we apply Active Learning (AL) in order to label those samples which are most relevant for increasing performance, with the aim of reducing the amount of labelled samples needed. We consider these annotated samples in both stages, the data generation part as well as the MRQA model. In our experiments we observe large improvements for the MRQA task for two domain-specific datasets, namely TechQA (Castelli et al., 2019) and BioASQ (Tsatsaronis et al., 2015), with few samples annotated. Moreover, the performance in both cases outperforms the setting with a full set of annotated data.

Our main contributions are as follows: 1) We introduce a novel approach that combines MRQA data augmentation via question-answer generation with Active Learning. 2) We identify the most relevant samples for AL by adapting to our setting scoring functions recently used for unsupervised quality assessment of machine translation. 3) We introduce a new sample relevance score specific to data generation by coupling the generated samples with the eventual task so that the MRQA model influences the data augmentation process. 4) We perform extensive experiments[1] to demonstrate how to best utilize data augmentation via data generation to boost performance in low-resource, domain-specific settings.

2 Related Work

2.1 Low-Resource MRQA

There are several approaches dealing with low-resource tasks, including settings where few labelled data are available and others where no labels are available at all. One approach is to use pre-trained language models (LM) (Alberti et al., 2019; Radford et al., 2019; Lewis et al., 2019), which can be especially useful in cases where little or no labelled data exists and it is costly to generate more. In the best case, the LM can be used without further fine-tuning. Otherwise, if unlabelled data is available, it may be used for adaptation of the LM in a self-supervised fashion.

If the low-resource domain is accompanied by some annotated samples, weak supervision – where labelled data is used as prior knowledge – is relevant (Hedderich et al., 2021; Wang et al., 2019). Moreover, data augmentation (Zhang et al., 2020; Van et al., 2021) and LM domain adaptation (Nishida et al., 2020; Zhang et al., 2020) have been shown to improve performance for the MRQA task. Another approach is to use domain transfer, where a model is trained on a source domain and then adapted to a different target domain e.g. by employing adversarial training (Lee et al., 2019). Last but not least, the use of AL for MRQA (Hong et al., 2019) has been shown to be helpful as well. However till date, it is limited to the model eventually used in the target domain (Hong et al., 2019).

2.2 Data Generation

Recent work has shifted domain adaptation to a data generation approach (Shakeri et al., 2020; Alberti et al., 2019; Puri et al., 2020; Luo et al., 2021; Lee et al., 2020). In this approach, given a passage of text, the generator model learns to output question-answer pairs. The generation model is trained on a large amount of data from the source domain and can be applied to any document in any target domain, commonly employing pre-trained transformers (Vaswani et al., 2017) as decoder.

Common generation methods focus only on generating the question and assess their performance exclusively on the generated question using automatic metrics (Sun et al., 2018; Liu et al., 2020; Tuan et al., 2019; Mitkov and Ha, 2003; Ma et al., 2020; Song et al., 2018; Duan et al., 2017; Yin et al., 2021; Sachan and Xing, 2018; Chen et al., 2020; Tang et al., 2017; Zhao et al., 2018; Du et al., 2017). Only a few approaches work to generate full question-answer pairs, where the generated data is evaluated by means of the MRQA model. While Klein and Nabi (2019) and Luo et al. (2021) only perform in-domain experiments – where the training data and data used for generating new samples comes from the same domain – Shakeri et al. (2020) and Lee et al. (2020) show the generalizability of their data generation models by also performing out of domain experiments. Furthermore, automatically generated question-answer pairs can be filtered (Alberti et al., 2019; Puri et al., 2020; Shakeri et al., 2020) to get rid of noisy samples.

3 Method

We focus on two main approaches relevant for the low-resource, domain-specific setting. The first one solely uses data augmentation with two different sample filtering scores. The second one is our novel approach that addresses the MRQA task by combining data generation with minimal human input in an AL setup. The sample selection in the

[1]The implementation for the experiments can be found here: https://www.removed-for-anonymity.edu/

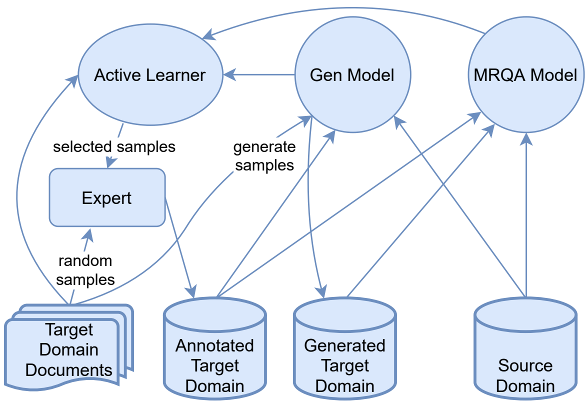


Figure 1: An overview of our approach: Samples for annotation are either chosen randomly or selected via the AL procedure, which takes the current models into account. The target domain is annotated by a data generation model which outputs question-answer pairs given a context.

AL setting is done by adapting and combining various existing scores, with the addition of one novel score. A high-level overview of our approach is provided in figure 1.

3.1 MRQA Model

We employ pre-trained BERT (Devlin et al., 2019) for encoding the question concatenated with the context. On top, a span extraction head models the probability for each context token being the start and the end of the answer span.

3.2 Data Generation for MRQA

Model We denote the question as q, the answer as a and the corresponding context as c. For the data generation model we use the QA2S model proposed by Shakeri et al. (2020) because this model shows best overall performance in their and our experiments. It makes use of a pre-trained encoder-decoder LM and it is fine-tuned to generate the question given the context, followed by generating the answer given the context and the previously decoded question in a subsequent decoding step. Hence the same model is trained to approximate probability distributions $p(a|q,c)$ and $p(q|c)$. Therefore, the question is generated in an auto-regressive manner and conditioned on the context:

$$p(q|c) = \sum_{t=1}^{T} \log p(q_t|q_{<t}, c). \qquad (1)$$

Similarly, the answer is conditioned on the context and the question:

$$p(a|q,c) = \sum_{t=1}^{T} \log p(a_t|a_{<t}, q, c). \qquad (2)$$

Special tokens <a>, and <q>, </q> mark the answer and the question in the sequence, respectively. During inference, a two-step decoding process is used and <a> and <q> are given as begin-of-sequence tokens to start generating the answer and the question, respectively.

Decoding Generating questions in the first step is done by nucleus sampling (Holtzman et al., 2020) from the output distribution over the vocabulary considering 95% of the probability mass and top 20 tokens in each step. The answer is then decoded greedily with a beam search of size 10. Samples for which the answer does not occur in the context are discarded. Furthermore, only samples for which the corresponding end tokens are predicted correctly are considered as valid.

Filtering Finally, only a subset of the generated samples are kept by using two sample filtering approaches, namely the LM score filtering introduced by Shakeri et al. (2020) and round-trip consistency (RTcons) (Alberti et al., 2019). In LM score filtering, the generated samples are sorted according to the probability $p(y|y_{<t}, x)$ as given by the generation model and the top n samples are kept (we use $n = 5$). For RTcons, an MRQA model is used to assess a generated question-answer pair: The generated question along with the context is fed into an MRQA model to predict the answer. The sample is discarded if the predicted answer does not match the generated one. Because RTcons might discard all samples generated from a context, we also fine-tune the MRQA model (pre-trained on the source domain) used for RTcons on the target domain if samples exist in the target domain.

3.3 Active Learning

AL is an iterative technique applied to reduce the amount of annotated samples needed for training while at the same time reaching high quality performance. At its core, a human with expert knowledge of the domain annotates selected samples. We implemented different sequence scoring functions in order to use them in an AL scenario for our specific task. While we borrow Sentence Probability (SP), Dropout-based Sentence Probability (D-SP) and Dropout-based Lexical Similarity (LS) from

Algorithm 1 Model Training with AL

```
Require:
    data 𝒟_pool, 𝒟_eval
    scoring function f_score
    iterations r, num samples n
initialize models ℳ_Gen, ℳ_MRQA with pre-trained weights
𝒟_train = ∅
for iteration = 1, 2, ..., r do
    𝒟_selected ← select n top scoring samples using f_score and ℳ_Gen, ℳ_MRQA for labelling
    𝒟_train ← 𝒟_train ∪ 𝒟_selected
    𝒟_pool ← 𝒟_pool \ 𝒟_selected
    re-initialize models ℳ_Gen, ℳ_MRQA with pre-trained weights
    train ℳ_Gen, ℳ_MRQA on 𝒟_train and load best models evaluated on 𝒟_eval (e.g. loss, F1)
end for
```

unsupervised quality estimates (Fomicheva et al., 2020; Xiao et al., 2020) and adapt them as scoring functions in our setting, we also introduce round-trip scoring (RT) that utilizes the MRQA model in the sample selection process. We believe that, during data generation, it is important to link the sample selection with the eventual downstream task in order to generate high quality samples. Finally, we also compare scoring functions based on the data generation model with an approach that directly uses the MRQA model. For this purpose we use Bayesian Active Learning by Disagreement (BALD) (Houlsby et al., 2011; Gal et al., 2017). Our novel approach combining data generation with AL is shown in Algorithm 1.

3.3.1 Scoring functions for generation model

Sentence Probability This scoring function makes use of the probability distribution of the data generation model. We score the contexts according to the sentence probability of the answer being generated:

$$\mathrm{SP}(\theta) = \frac{1}{T}\sum_{t=1}^{T} \log p(a_t | a_{<t}, c, q, \theta) \quad (3)$$

We generate the question and answer by decoding with beam search of size 10 and greedy decoding, but do only use the produced answer for scoring a sample's context.

Dropout-based Sentence Probability In contrast to SP, D-SP makes use of multiple data generation models to compute the sentence probability:

$$\text{D-SP} = \frac{1}{N}\sum_{n=1}^{N} \mathrm{SP}(\theta_n) \quad (4)$$

We employ dropout at inference time (as well as during training) to realize different subsets of the model, as described by Gal and Ghahramani (2016), running N forward passes. Note that the output is decoded similarly as in SP, and the same prediction is used in all forward passes.

Dropout-based Lexical Similarity For this scoring function, multiple answers are decoded using multiple models (again realized via dropout), and all of them are compared pairwise at the lexical level (with $i \neq j$):

$$\mathrm{LS} = \frac{1}{N*(N-1)}\sum_{i=1}^{N}\sum_{j=1}^{N} \mathrm{Meteor}(a_i, a_j) \quad (5)$$

Decoding is again implemented greedily using beam search of size 10.

Because the ultimate goal is to improve the MRQA model, we propose the following novel methods that integrate the MRQA model in the computation of the ranking scores:

Round-trip (RT) For each generated question-answer pair from the given context (generation is done similar to SP), we apply an MRQA model on it and compute the F1 score between the predicted answer and the answer generated by the data generation model and use it to rank the samples.

D-SP+RT We combine the D-SP and the RT scores sample-wise to obtain the final ranking score. We rescale the D-SP score such that the best prediction is assigned 1.0 (i.e. the ranges match the RT score) and distribution is similar to RT:

$$\text{D-SP+RT} = \exp(4 \times \text{D-SP})^2 + \mathrm{RT} \quad (6)$$

3.3.2 Scoring function for MRQA model

In case of only using the MRQA model, we use the BALD score (Houlsby et al., 2011) to measure the uncertainty of the model's predictions over training data $\mathcal{D}$ and model weights θ using entropy H:

$$\mathrm{H}[y|x,\mathcal{D}] - \mathbb{E}_{\theta\sim p(\theta|\mathcal{D})}\mathrm{H}[y|x,\theta] \quad (7)$$

In order to make computation feasible we calculate the BALD score (using dropout) independently for start and end probabilities. The sum of both scores is then used to order and rate the samples.

4 Experimental Setup

In our experiments we consider three different low-resource settings: 1) only unlabelled data from the target domain (S1), 2) statically labelled data from the target domain (S2) as well as 3) dynamically labelled data from the target domain (S3).

4.1 Data

We use SQuAD as source domain and NaturalQuestions (NQ), TechQA and BioASQ as target domains (some statistics are summarized in table 1). The main reason for choosing these datasets (and their sampled low-resource versions) is to have examples of two very specialized domains (TechQA, BioASQ) completely different from the source domain, as well as an example of a domain overlapping with the source domain (NQ). In addition to each full domain we create two low-resource scenarios, one by choosing 200 samples randomly (denoted as NQ-200, TechQA-200 and BioASQ-200) and the second via the AL procedure described in section 3.3, respectively. These correspond to our settings where few training data is available (S2 and S3). More details can be found in the appendix.

Domain	context tokens			questions tokens			answer tokens		
	min	max	mean	min	max	mean	min	max	mean
SQuAD	25	853	155.75	1	61	12.29	1	68	4.23
NQ	10	3143	245.4	7	29	9.76	1	270	5.25
TechQA	38	38925	1484.08	6	561	69.37	5	545	98.76
BioASQ	27	960	337.71	5	36	15.06	1	76	4.42

Table 1: Statistics of the train split of the domains used in this work obtained with the tokenizer for BERT.

4.2 Question & Answer Generation

Training Similarly to Shakeri et al. (2020) we use `bart-large` (Lewis et al., 2019) in our data generation model. It contains 406M parameters and it is trained for 5 or 10 epochs depending on whether training is performed on the source domain or on the target domain only. We use cross entropy loss to train it on the source domain, and select the model with the best loss on the evaluation data. For the low-resource target domains we additionally experiment training the data generation model only on target data, as well as combined with the source domain. In the latter case we train the data generation model with samples from the source domain first and fine-tune on the target domain afterwards.

To fit the context concatenated with the question into the model we split the context into several chunks. In the training we only consider those chunks where the answer does occur in the context. Additionally, since TechQA has long questions, we truncate the questions to the first 200 tokens to allow for sufficient context in the input.

Synthetic Data Generation Regarding the question generation (i.e. the first decoding step), we allow contexts to have a size of up to 724 tokens so that the generated questions may comprise up to 300 tokens, as both will be fed into the model in the second decoding step. Chunking this input is difficult since aggregating generated text is not simple (especially if outputs do not overlap). We generate 10 questions per input followed by one answer for each question-context pair, yielding up to 10 samples per context. In total we choose 100000 documents from the target domain, selecting first 550 tokens and ignoring contexts with less than 100 tokens. For NQ, only 50535 documents remain.

4.3 MRQA Model

We use `bert-base-uncased` in the MRQA model with a maximum input length of 512 tokens and a stride of 128. As for the data generation model we similarly truncate questions in case of TechQA (whether from the train/dev corpus or synthetically generated. Since chunking is applied to the inputs and predictions are aggregated afterwards (by choosing the best span over all chunks' predictions), it may happen that the answer of a sample is not completely included in a chunk's context. We especially observe this for TechQA, where the performance on the evaluation data is greatly degraded as the model never has the chance to predict the answer correctly for some samples.

4.4 Active Learning

For the low-resource domains employing AL we score available samples using each of the scoring functions as described in Section 3.3, running 10 forward passes for D-SP, LS, D-SP+RT and BALD. We run 4 iterations starting with all available samples from the target domain in the pool and select the 50 samples the model is least confident about. In each iteration we remove the selected samples from the query pool and train the models on all selected samples so far, always starting with the model trained on the source domain (or synthetic data for MRQA). Since D-SP, LS and RT are computationally expensive, with LS actually decoding generated text after each forward pass, performing AL with LS, RT and D-SP+RT on NQ was not feasible for us. Hence we randomly selected 10000 samples from this domain for use with AL, denoted as NQ#10000 in our experiments.

Setting	Fine-tune data for generation model	Fine-tune data for MRQA model	Filtering	NQ		NQ-200		TechQA		TechQA-200		BioASQ		BioASQ-200	
				EM	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
S1	-	SQuAD	-	43.42	58.12	43.42	58.12	0.63	17.06	0.63	17.06	48.50	62.99	48.50	62.99
S2	-	SQuAD + Target	-	67.53 ± 0.24	79.31 ± 0.19	49.26 ± 0.23	62.72 ± 0.3	33.13 ± 1.43	60.30 ± 0.32	27.75 ± 0.64	56.19 ± 0.89	78.21 ± 1.02	82.77 ± 0.72	67.31 ± 0.72	73.96 ± 0.32
S2*	-	Target	-	66.63	78.64	22.94	32.24	26.88	55.9	21.88	52.45	78.07	80.64	49.50	54.54
S1	SQuAD	Synthetic	LM	50.72	63.8	50.72	63.8	0.63	19.76	0.63	19.76	50.5	62.99	50.5	62.99
	SQuAD	Synthetic	RTcons	51.37	64.38	51.37	64.38	1.88	16.34	1.88	16.34	50.17	62.92	50.17	62.92
S2	SQuAD	Synthetic + Target	LM	**67.63 ± 0.12**	**79.45 ± 0.03**	52.79 ± 0.19	65.51 ± 0.19	34.13 ± 0.94	60.42 ± 0.49	27.75 ± 0.75	**57.12 ± 1.04**	82.46 ± 0.88	87.08 ± 0.53	69.24 ± 0.45	76.65 ± 0.48
	SQuAD	Synthetic + Target	RTcons	67.39 ± 0.21	79.34 ± 0.18	52.35 ± 0.08	65.33 ± 0.1	34.13 ± 1.29	61.33 ± 1.31	28.25 ± 0.92	56.90 ± 1.08	81 ± 0.71	85.92 ± 0.32	64.85 ± 0.71	72.66 ± 0.80
	Target	Synthetic	LM	-	-	44.62	55.5	15.63	42.22	11.25	35.87	76.41	82.35	62.79	69.37
	Target	Synthetic	RTcons	-	-	49.92	61.23	21.88	48.68	20.63	46.55	70.1	77.95	65.78	72.64
	Target	Synthetic + Target	LM	-	-	46.82 ± 0.16	59.1 ± 0.19	31.88 ± 1.05	58.78 ± 0.6	**30.75 ± 1.08**	57.11 ± 0.89	84.92 ± 0.5	87.66 ± 0.6	73.36 ± 1.06	78.14 ± 1.01
	Target	Synthetic + Target	RTcons	-	-	50.09 ± 0.12	61.92 ± 0.1	33.88 ± 0.47	60.88 ± 0.71	30.63 ± 0.4	55.9 ± 0.87	83.52 ± 0.16	87.01 ± 0.20	**73.36 ± 0.64**	77.96 ± 0.53
	SQuAD + Target	Synthetic	LM	-	-	53.96	66.4	17.5	39.88	17.5	36.5	75.08	83.74	70.1	77.24
	SQuAD + Target	Synthetic	RTcons	-	-	54.67	66.98	24.38	43.93	23.13	41.36	76.08	82.08	68.44	76.15
	SQuAD + Target	Synthetic + Target	LM	-	-	54.61 ± 0.15	**67.13 ± 0.15**	37 ± 0.73	**62.78 ± 0.44**	30.63 ± 0.83	57.04 ± 1.24	**85.51 ± 0.58**	**88.95 ± 0.35**	72.36 ± 0.95	**80.15 ± 0.85**
	SQuAD + Target	Synthetic + Target	RTcons	-	-	**54.67 ± 0.13**	67.12 ± 0.14	**38.5 ± 1.51**	62.21 ± 1.15	30.25 ± 0.75	56.58 ± 1.06	84.25 ± 0.5	88.07 ± 0.31	70.83 ± 0.49	78.35 ± 0.50

Table 2: EM and F1 scores (with standard deviation across 5 runs where indicated) on the dev sets for the downstream MRQA task for NaturalQuestions, TechQA and BioASQ, as well as their low-resource versions with 200 samples drawn randomly. In contrast to NQ we also consider TechQA and BioASQ as low-resource domain, with only 450 and 1052 training samples, respectively. The results for training the MRQA model on the synthetic data when SQuAD is available only are the same for the target domains and their respective low-resource variants, as no target domain data was used for training, neither in the data generation nor in the MRQA model. Best results per domain (excluding AL) are marked bold. S1 considers only the source domain for training, whereas S2 adds few samples from the target domain.
* Supervised target domain

5 Experimental Results

5.1 Unlabelled target domain & low-resource samples drawn randomly

In our experiments (cf. table 2) we observe that SQuAD cannot be used in the first setting (S1) directly (i.e. without synthetic data generation) on TechQA (F1 of 17.06% vs. 55.9% (TechQA) and 52.45% (TechQA-200)). However, with labelled target domain data available in addition to SQuAD, performance increases to 60.3% F1 on TechQA and 56.19% F1 on TechQA-200. Regarding NQ cast as low-resource domain, performance drops (32.24% F1) with supervised in-domain data only and, without any synthetic data, SQuAD already improves NQ-200 (58.12% F1). We see further gain of 4.6% F1 if the MRQA model is fine-tuned on NQ-200. With the same amount of samples in the BioASQ domain the performance of the MRQA model increases from 54.54% F1 if trained on in-domain data only to 62.99% if only SQuAD data is used. Fine-tuning on BioASQ data yields 73.96% F1 (BioASQ-200) and 82.77% F1 (BioASQ).

When only unlabelled data is available in the target domain we see small gains (+2.7% F1) for TechQA when training the MRQA model on synthetic data if only SQuAD is used (in training the data generation model). LM filtering works better than RTcons filtering likely for the same reason that the downstream MRQA model trained on SQuAD does not perform well on TechQA and a domain independent filtering avoids this issue. In the case of NQ-200, RTcons filtering works best, yielding 64.38% F1, which is supported by Shakeri et al. (2020) showing the same trend. With this setup we do not observe any benefit for BioASQ.

Comparing performance for TechQA and TechQA-200 on synthetic data only in S2 (that means with few labelled target data available), the data generation model trained only on target data (thus omitting the source domain at all) shows a better performance (48.68% F1 vs. 43.93% F1 (TechQA) and 46.55% F1 vs. 41.36% F1 (TechQA-200)). However, if the MRQA model is also fine-tuned on the target domain data, we get best results with 62.78% F1 (TechQA) and 57.12% F1 (TechQA-200) if data generation is trained on SQuAD (in case of TechQA additionally to target data). In contrast we consistently get better results for NQ-200 if SQuAD is used as well. The low-resource version of NQ performs best if both the data generation and the MRQA model are fine-tuned on its samples (67.13% F1), but there is no significant difference whether target data is used in fine-tuning the MRQA model or not. We consistently get large gains and best results for the TechQA domains if the MRQA model is further fine-tuned on target domain data. Similarly, we observe best performance for BioASQ-200 if the MRQA model is fine-tuned on the target domain with a score of 80.15% F1 (fine-tuning the data generation model on the target domain data as well).

5.2 AL for Data Generation & MRQA

In total, as can be seen in table 3, AL improves MRQA for all domains. Our newly introduced scoring function RT and its extension D-SP+RT together outperform the other AL approaches on two domains, TechQA (58.89% F1) and NQ#10000

Setup	AL strategy	Fine-tune data for MRQA model	Filtering	NQ EM	NQ F1	NQ#10000 EM	NQ#10000 F1	TechQA EM	TechQA F1	BioASQ EM	BioASQ F1
Gen*	SP	Synthetic	LM	53.45	66.24	54	66.51	13.75	35.51	65.78	75.27
	SP	Synthetic	RTcons	54.49	67.24	55.31	67.94	21.88	41.98	67.11	76.52
	SP	Synthetic + Target	LM	54.30 ± 0.18	66.89 ± 0.15	54.48 ± 0.2	67.32 ± 0.18	30.38 ± 0.85	56.51 ± 0.73	72.03 ± 0.77	80.63 ± 0.59
	SP	Synthetic + Target	RTcons	54.48 ± 0.02	67.23 ± 0	55.39 ± 0.07	68.04 ± 0.07	31 ± 0.75	56.43 ± 0.7	68.77 ± 0.66	77.2 ± 0.52
	D-SP	Synthetic	LM	53.96	66.57	54.53	67.02	16.25	36.69	68.11	77.21
	D-SP	Synthetic	RTcons	54.68	67.23	54.82	67.11	28.13	46.56	66.11	74.91
	D-SP	Synthetic + Target	LM	54.40 ± 0.23	67.13 ± 0.14	55.11 ± 0.08	67.86 ± 0.08	30 ± 0.88	55.64 ± 0.44	**75.35 ± 0.64**	**82.57 ± 0.52**
	D-SP	Synthetic + Target	RTcons	**54.80 ± 0.16**	**67.68 ± 0.11**	54.98 ± 0.03	67.50 ± 0.08	**32.5 ± 0.56**	56.17 ± 0.41	72.56 ± 0.68	80.06 ± 0.75
	LS	Synthetic	LM	-	-	54.53	67.11	18.13	37.63	66.45	75.53
	LS	Synthetic	RTcons	-	-	54.78	67.61	23.75	43.31	66.45	74.93
	LS	Synthetic + Target	LM	-	-	54.83 ± 0.19	67.75 ± 0.16	30.63 ± 0.79	55.42 ± 0.47	68.64 ± 0.4	79.06 ± 0.24
	LS	Synthetic + Target	RTcons	-	-	54.78 ± 0	67.61 ± 0	31.25 ± 1.25	55.5 ± 1.39	71.89 ± 0.72	79.95 ± 0.53
MRQA**	RT	Synthetic	LM	-	-	54.5	66.64	16.88	37.32	67.77	76.76
	RT	Synthetic	RTcons	-	-	54.85	67.48	23.13	45.97	65.12	74.25
	RT	Synthetic + Target	LM	-	-	55.43 ± 0.05	**68.28 ± 0.08**	31.5 ± 1.09	56.85 ± 0.53	71.96 ± 0.16	80.23 ± 0.44
	RT	Synthetic + Target	RTcons	-	-	55 ± 0.18	67.71 ± 0.14	32 ± 0.47	**58.89 ± 0.62**	70.56 ± 0.95	79.17 ± 0.86
	D-SP+RT	Synthetic	LM	-	-	54.88	66.86	18.13	35.69	67.11	75.63
	D-SP+RT	Synthetic	RTcons	-	-	54.25	66.69	23.13	45.03	66.78	75.35
	D-SP+RT	Synthetic + Target	LM	-	-	**55.55 ± 0.14**	67.77 ± 0.15	28.38 ± 1.16	53.84 ± 0.39	71.69 ± 1.08	78.77 ± 1.13
	D-SP+RT	Synthetic + Target	RTcons	-	-	54.65 ± 0.08	67.50 ± 0.08	30.75 ± 0.73	57.81 ± 0.36	67.97 ± 0.45	76.39 ± 0.42
MRQA-only***	BALD	Synthetic + Target	LM	52.49 ± 0.43	66.05 ± 0.41	53.10 ± 0.47	66.58 ± 0.42	29.25 ± 0.25	56.12 ± 1.06	68.7 ± 1.02	77.44 ± 0.53
	BALD	Synthetic + Target	RTcons	52.53 ± 0.44	66.18 ± 0.36	52.74 ± 0.48	66.39 ± 0.24	27.88 ± 1.29	57.04 ± 0.93	65.18 ± 1.75	74.15 ± 1.28

Table 3: EM and F1 scores (with standard deviation across 5 runs where indicated) for AL on target domain (setting S3) where target refers to the samples selected via AL (4 iterations with 50 samples queried). In all cases the data generation model is trained on SQuAD and fine-tuned on target domain data. AL improves results in all low-resource settings, and consistently improves NQ and NQ#10000 where many samples are available to be queried. Best overall results for low-resource domains are marked bold.

* makes only use of the data generation model during scoring and trains the data generation model on the newly annotated samples followed by synthetic data generation.

** like Gen, but makes use of the MRQA model during scoring.

*** data generation model was trained on SQuAD data only, hence the MRQA pre-trained model from table 2 (first two synthetic rows) has been used (i.e the best models without target domain data).

(68.28% F1). Although RT scoring also performs better on BioASQ than random sampling, we get the best results by using D-SP (82.57%).

BALD performs better than random sampling in terms of F1 score if applied to the MRQA model pre-trained on synthetic data except for TechQA (for which the F1 score is lower but the EM score higher). However, as expected, it performs worse than if data generation is employed for the MRQA task, highlighting the benefit of data augmentation for the MRQA task and the application of AL at the data generation model.

	BALD	SP	LS	RT	D-SP+RT	D-SP	RANDOM
BALD	200	39	44	42	54	49	42
SP	39	200	49	49	66	72	40
LS	44	49	200	37	48	46	41
RT	42	49	37	200	63	49	15
D-SP+RT	54	66	48	63	200	82	42
D-SP	49	72	46	49	82	200	36
RANDOM	42	40	41	15	42	36	200

Table 4: Overlap of samples for the various AL approaches on BioASQ: The overlap of the scoring functions is rather small.

6 Insights and Lessons Learned

6.1 Insights of drawn samples

To better understand the performance of the considered AL methods, we have analyzed different aspects of the samples selected for labelling.

Overlap of drawn samples We analyzed the overlap of the chosen samples between the different AL strategies. Table 4 reflects this for the BioASQ domain. We observe rather small sample overlap among the approaches, a trend that holds for the other domains as well.

Distribution of scores Figure 2 shows the distribution of scores over all available samples in each iteration of AL for TechQA. We observe a steep ascent with RT where, depending on the dataset, many samples are rated as 0 and many as 1. This indicates that both models, data generation and MRQA, work very well in combination. The RT scoring is also a good measure to quantify the difference between the target domain and the source domain. For example, for both NQ and BioASQ RT scores are much better than those for TechQA. This is probably because the RT method considers the downstream MRQA task when computing the score where the distribution shift is larger for more specialized domains. Therefore, for the TechQA dataset one might as well select all 200 samples in one iteration instead of annotating 50 samples in four consecutive iterations. Another interesting

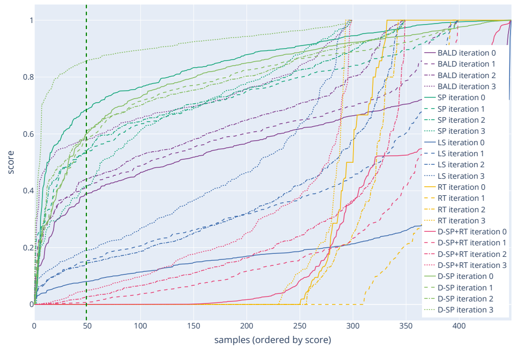


Figure 2: Sample score distribution for TechQA: RT scores many samples low, but surprisingly also rates some samples best although the task of predicting the generated answer for a generated question is complex. Scores have been rescaled to $[0, 1]$.

behavior we observe is that RT scores samples best in the first iteration. This might occur due to potential overfitting of the data generation model, being trained with 50 samples from the target domain.

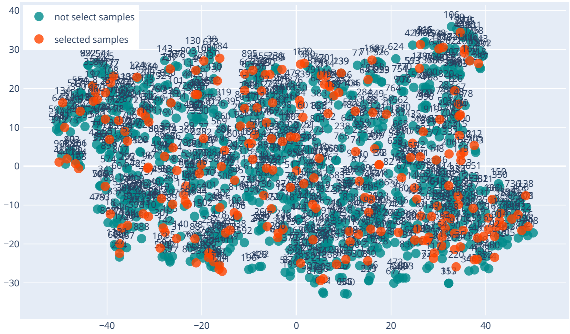


Figure 3: Sample encoding distribution from MRQA on BioASQ with RT for samples selected (221 instances) after the last iteration.

Distribution of samples In order for the MRQA model to perform well, it is important that a set of diverse samples is selected in the AL strategy. We visualize the diversity of the sample selection process for each AL strategy using t-SNE. Figure 3 visualizes samples drawn according to RT after the last iteration using BioASQ and shows their diversity. More examples can be found in the appendix.

6.2 Lessons Learned

S1: MRQA domain transfer, where no labelled data from the target domain is available, yields poor performance for specialized domains. This can be very well observed for TechQA, with the domain being very different from the one of the source SQuAD dataset. In this scenario data generation also does not warrant an improvement, as showcased for the BioASQ dataset. In case of TechQA, very little improvement could be observed. This could be due to the style of the generated questions and the target domain questions differing too much.

S2: Adding labelled samples from the target domain already increases performance without data augmentation via generation, especially for very specialized domains. When a small amount of labelled samples from the target domain exists, generating synthetic data to augment the training set still offers a robust solution, independently from the similarity between source and target domains. We observe improved results for all target domains if SQuAD is used in training the data generation model. This underlines our hypothesis that data generation improves MRQA performance even if target domain training data is available as questions are asked on contexts from the target domain.

S3: For all domains, applying AL shows an improvement when compared to S1 and S2, especially in the data generation process. Furthermore, our proposed method - RT scoring - provides a competitive scoring function. F1 scores are consistently increased by 1.15% (NQ), 1.77% (TechQA) and 2.42% (BioASQ). We expect that this improvement could be further increased when more (unlabelled) samples are available for querying. This can thus be used as an approach to minimize the costly annotation effort in specialized domains.

7 Conclusion

Although data scarcity is known in practice, there is a lack of approaches that address this problem for the MRQA task in specialized domains. Therefore, realistic low-resource domains and appropriate methods are needed since deep learning models usually work well when plenty of annotated, in-domain data are available. In our work we demonstrated how to best utilize data augmentation via generation to boost performance when addressing the challenging MRQA problem in low-resource, domain-specific settings relevant in practice. We also introduced a novel approach that combines data generation with active learning when tackling the MRQA problem to enable performance boost with low labelling effort. To this end we assessed AL performance when applied to the MRQA model directly as well as in the process of generating data and showed significant predictive performance improvements, also when using our newly introduced scoring function tailored to the MRQA task.

References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019. Synthetic QA Corpora Generation with Roundtrip Consistency. *arXiv:1906.05416 [cs]*. ArXiv: 1906.05416.

Vittorio Castelli, Rishav Chakravarti, Saswati Dana, Anthony Ferritto, Radu Florian, Martin Franz, Dinesh Garg, Dinesh Khandelwal, Scott McCarley, Mike McCawley, Mohamed Nasr, Lin Pan, Cezar Pendus, John Pitrelli, Saurabh Pujar, Salim Roukos, Andrzej Sakrajda, Avirup Sil, Rosario Uceda-Sosa, Todd Ward, and Rong Zhang. 2019. The TechQA Dataset. *arXiv:1911.02984 [cs]*. ArXiv: 1911.02984.

Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2020. Reinforcement Learning Based Graph-to-Sequence Model for Natural Question Generation. *arXiv:1908.04942 [cs]*. ArXiv: 1908.04942.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805 [cs]*. ArXiv: 1810.04805.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to Ask: Neural Question Generation for Reading Comprehension. *arXiv:1705.00106 [cs]*. ArXiv: 1705.00106.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question Generation for Question Answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874, Copenhagen, Denmark. Association for Computational Linguistics.

Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. MRQA 2019 Shared Task: Evaluating Generalization in Reading Comprehension. *arXiv:1910.09753 [cs]*. ArXiv: 1910.09753.

Marina Fomicheva, Shuo Sun, Lisa Yankovskaya, Frédéric Blain, Francisco Guzmán, Mark Fishel, Nikolaos Aletras, Vishrav Chaudhary, and Lucia Specia. 2020. Unsupervised Quality Estimation for Neural Machine Translation. *arXiv:2005.10608 [cs]*. ArXiv: 2005.10608.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *arXiv:1506.02142 [cs, stat]*. ArXiv: 1506.02142.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep Bayesian Active Learning with Image Data. *arXiv:1703.02910 [cs, stat]*. ArXiv: 1703.02910.

Michael A. Hedderich, Lukas Lange, Heike Adel, Jannik Strötgen, and Dietrich Klakow. 2021. A Survey on Recent Approaches for Natural Language Processing in Low-Resource Scenarios. *arXiv:2010.12309 [cs]*. ArXiv: 2010.12309.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The Curious Case of Neural Text Degeneration. *arXiv:1904.09751 [cs]*. ArXiv: 1904.09751.

Yining Hong, Jialu Wang, Yuting Jia, Weinan Zhang, and Xinbing Wang. 2019. Academic Reader: An Interactive Question Answering System on Academic Literatures. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):9855–9856. Number: 01.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. 2011. Bayesian Active Learning for Classification and Preference Learning. *arXiv:1112.5745 [cs, stat]*. ArXiv: 1112.5745.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*. ArXiv: 1412.6980.

Tassilo Klein and Moin Nabi. 2019. Learning to Answer by Learning to Ask: Getting the Best of GPT-2 and BERT Worlds. *arXiv:1911.02365 [cs]*. ArXiv: 1911.02365.

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. 2019. Natural Questions: a Benchmark for Question Answering Research. *Transactions of the Association of Computational Linguistics*.

Dong Bok Lee, Seanie Lee, Woo Tae Jeong, Donghwan Kim, and Sung Ju Hwang. 2020. Generating Diverse and Consistent QA pairs from Contexts with Information-Maximizing Hierarchical Conditional VAEs. *arXiv:2005.13837 [cs]*. ArXiv: 2005.13837.

Seanie Lee, Donggyu Kim, and Jangwon Park. 2019. Domain-agnostic Question-Answering with Adversarial Training. *arXiv:1910.09342 [cs]*. ArXiv: 1910.09342.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *arXiv:1910.13461 [cs, stat]*. ArXiv: 1910.13461.

Quentin Lhoest, Albert Villanova del Moral, Patrick von Platen, Thomas Wolf, Yacine Jernite, Abhishek Thakur, Lewis Tunstall, Suraj Patil, Mariama Drame, Julien Chaumond, Julien Plu, Joe Davison, Simon Brandeis, Teven Le Scao, Victor Sanh, Kevin Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Steven Liu, Sylvain Lesage, Lysandre Debut, Théo Matussière, Clément Delangue, and Stas Bekman. 2021. huggingface/datasets: 1.11.0.

Bang Liu, Haojie Wei, Di Niu, Haolan Chen, and Yancheng He. 2020. Asking Questions the Human Way: Scalable Question-Answer Generation from Text Corpus. *Proceedings of The Web Conference 2020*, pages 2032–2043. ArXiv: 2002.00748.

Hongyin Luo, Shang-Wen Li, Seunghak Yu, and James Glass. 2021. Cooperative Learning of Zero-Shot Machine Reading Comprehension. *arXiv:2103.07449 [cs]*. ArXiv: 2103.07449.

Xiyao Ma, Qile Zhu, Yanlin Zhou, Xiaolin Li, and Dapeng Wu. 2020. Improving Question Generation with Sentence-level Semantic Matching and Answer Position Inferring. *arXiv:1912.00879 [cs]*. ArXiv: 1912.00879.

Ruslan Mitkov and Le An Ha. 2003. Computer-aided generation of multiple-choice tests. In *Proceedings of the HLT-NAACL 03 workshop on Building educational applications using natural language processing - Volume 2*, HLT-NAACL-EDUC '03, pages 17–22, USA. Association for Computational Linguistics.

Kosuke Nishida, Kyosuke Nishida, Itsumi Saito, Hisako Asano, and Junji Tomita. 2020. Unsupervised Domain Adaptation of Language Models for Reading Comprehension. *arXiv:1911.10768 [cs]*. ArXiv: 1911.10768.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch.

Raul Puri, Ryan Spring, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2020. Training Question Answering Models From Synthetic Data. *arXiv:2002.09599 [cs]*. ArXiv: 2002.09599.

Alec Radford, Jeff Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. *undefined*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv:1606.05250 [cs]*. ArXiv: 1606.05250.

Mrinmaya Sachan and Eric Xing. 2018. Self-Training for Jointly Learning to Ask and Answer Questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 629–640, New Orleans, Louisiana. Association for Computational Linguistics.

Siamak Shakeri, Cicero Nogueira dos Santos, Henry Zhu, Patrick Ng, Feng Nan, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2020. End-to-End Synthetic Data Generation for Domain Adaptation of Question Answering Systems. *arXiv:2010.06028 [cs]*. ArXiv: 2010.06028.

Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. 2018. Leveraging Context Information for Natural Question Generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 569–574, New Orleans, Louisiana. Association for Computational Linguistics.

Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and Position-aware Neural Question Generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3930–3939, Brussels, Belgium. Association for Computational Linguistics.

Duyu Tang, Nan Duan, Tao Qin, Zhao Yan, and Ming Zhou. 2017. Question Answering and Question Generation as Dual Tasks. *arXiv:1706.02027 [cs]*. ArXiv: 1706.02027.

George Tsatsaronis, Georgios Balikas, Prodromos Malakasiotis, Ioannis Partalas, Matthias Zschunke, Michael R. Alvers, Dirk Weissenborn, Anastasia Krithara, Sergios Petridis, Dimitris Polychronopoulos, Yannis Almirantis, John Pavlopoulos, Nicolas Baskiotis, Patrick Gallinari, Thierry Artiéres, Axel-Cyrille Ngonga Ngomo, Norman Heino, Eric Gaussier, Liliana Barrio-Alvers, Michael Schroeder, Ion Androutsopoulos, and Georgios Paliouras. 2015. An overview of the BIOASQ large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16(1):138.

Luu Anh Tuan, Darsh J. Shah, and Regina Barzilay. 2019. Capturing Greater Context for Question Generation. *arXiv:1910.10274 [cs]*. ArXiv: 1910.10274.

Hoang Van, Vikas Yadav, and Mihai Surdeanu. 2021. Cheap and Good? Simple and Effective Data Augmentation for Low Resource Machine Reading. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2116–2120. ArXiv: 2106.04134.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv:1706.03762 [cs]*. ArXiv: 1706.03762.

Yaqing Wang, Quanming Yao, James Kwok, and Lionel M. Ni. 2019. Generalizing from a Few Examples: A Survey on Few-Shot Learning. *arXiv:1904.05046 [cs]*. ArXiv: 1904.05046.

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. 2020. Transformers: State-of-the-art natural language processing. In

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 38–45, Online. Association for Computational Linguistics.

Tim Z. Xiao, Aidan N. Gomez, and Yarin Gal. 2020. Wat zei je? Detecting Out-of-Distribution Translations with Variational Transformers. *arXiv:2006.08344 [cs, stat]*. ArXiv: 2006.08344.

Xusen Yin, Li Zhou, Kevin Small, and Jonathan May. 2021. Summary-Oriented Question Generation for Informational Queries. *arXiv:2010.09692 [cs]*. ArXiv: 2010.09692.

Rong Zhang, Revanth Gangi Reddy, Md Arafat Sultan, Vittorio Castelli, Anthony Ferritto, Radu Florian, Efsun Sarioglu Kayi, Salim Roukos, Avirup Sil, and Todd Ward. 2020. Multi-Stage Pre-training for Low-Resource Domain Adaptation. *arXiv:2010.05904 [cs]*. ArXiv: 2010.05904.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level Neural Question Generation with Maxout Pointer and Gated Self-attention Networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910, Brussels, Belgium. Association for Computational Linguistics.

A Data

SQuAD We use the official SQuAD 1.1 dataset (Rajpurkar et al., 2016) and its split in training and development sets comprised of 87599 and and 10570 samples, respectively. We use SQuAD as source domain for training the data generation model and for training the MRQA model in the in-domain setting, as well as together with the target data in case of the low-resource scenarios.

NaturalQuestions We use the NaturalQuestions dataset (Kwiatkowski et al., 2019), preprocessed for the MRQA Shared Task 2019 (Fisch et al., 2019). The train split contains 104071 samples while the dev split has 12836 samples. In our work this dataset is used due to its domain being similar to the one of SQuAD. The documents from the training set are used to generate question-answer pairs while documents from the dev set are excluded.

TechQA TechQA is a dataset released by IBM (Castelli et al., 2019). Questions are asked in a forum post style and the answer is given as spans within technotes (i.e. the documents). It contains 600 training samples including unanswerable questions. In our work we only consider the answerable subset of 450 questions. The official dev data with 160 answerable samples was used for evaluation since the test set is kept secret to ensure integrity of the benchmark. Passages from the corpus containing 800K+ documents were used to generate data.

BioASQ Regarding the BioASQ dataset (Tsatsaronis et al., 2015), we rely on the version preprocessed for the MRQA Shared Task 2019 (Fisch et al., 2019). It includes a dev set with 1504 samples. We create random splits for training, development and testing with 70%, 20% and 10% of samples, respectively. For the data generation process we crawl PubMed abstracts as unlabelled passages.

B Implementational Details

In our implementation we rely on PyTorch (Paszke et al., 2017) and the transformers library (Wolf et al., 2020) for the models as well as training. The datasets library (Lhoest et al., 2021) is used for loading and preprocessing data.

B.1 Hyperparameters

The MRQA as well as the data generation model have been trained with Adam Optimizer (Kingma and Ba, 2017), a learning rate of $3e{-}5$ and a batch size of 24. Warm-up was set to 10% for training the data generation while it was disabled in case of MRQA training. Similar to the models used for the downstream MRQA task `bert-base-uncased` was used for RTcons filtering.

MRQA model training turned out to be quite fluctuating in terms of the evaluation score. Hence we performed a hyperparameter search including learning rate, warm-up steps (using linear scheduler), L2 regularization, pre-trained weight decay for the encoder and for the output layer separately, freezing the encoder or its embedding, training of only top-n layers and re-initializing top-n layers. As a result, only pre-trained weight decay on the encoder with $\lambda = 1e{-}7$ was employed while all layers were trained without re-initialization.

C Further Analysis

Tables 5 and 6 show the statistics and sample overlap for all domains.

Figure 4 shows the distribution of scores over all queryable samples in each iteration of AL for the different datasets.

Figure 5 shows the sample distribution for BioASQ using RT scoring for AL.

	# samples	# unique contexts	# instances (rc / qa2s)
BALD	200	194	269 / 217
SP	200	161	807 / 405
D-SP	200	181	619 / 338
RANDOM	200	200	240 / 213

(a) NQ

	# samples	# unique contexts	# instances (rc / qa2s)
BALD	200	200	311 / 229
SP	200	185	746 / 377
LS	200	199	403 / 262
RT	200	197	436 / 273
D-SP+RT	200	195	643 / 340
D-SP	200	192	654 / 353
RANDOM	200	200	240 / 213

(b) NQ#10000

	# samples	# unique contexts	# instances (rc / qa2s)
BALD	200	177	909 / 513
SP	200	175	1635 / 851
LS	200	177	1597 / 832
RT	200	182	1185 / 643
D-SP+RT	200	181	1595 / 847
D-SP	200	176	1614 / 855
RANDOM	200	182	1034 / 557

(c) TechQA

	# samples	# unique contexts	# instances (rc / qa2s)
BALD	200	199	236 / 200
SP	200	200	225 / 200
LS	200	200	220 / 200
RT	200	199	221 / 200
D-SP+RT	200	197	222 / 200
D-SP	200	199	227 / 200
RANDOM	200	199	216 / 200

(d) BioASQ

Table 5: Statistics for the various domains: Although the amount of unique contexts fluctuates between the AL strategies, there is no correlation with the performance on the MRQA model.

	BALD	SP	D-SP	RANDOM
BALD	200	0	0	0
SP	0	200	2	1
D-SP	0	2	200	0
RANDOM	0	1	0	200

(a) NQ

	BALD	SP	LS	RT	D-SP+RT	D-SP	RANDOM
BALD	200	8	11	3	8	6	0
SP	8	200	20	18	38	42	6
LS	11	20	200	7	20	21	5
RT	3	18	7	200	17	10	0
D-SP+RT	8	38	20	17	200	70	3
D-SP	6	42	21	10	70	200	5
RANDOM	0	6	5	0	3	5	200

(b) NQ#10000

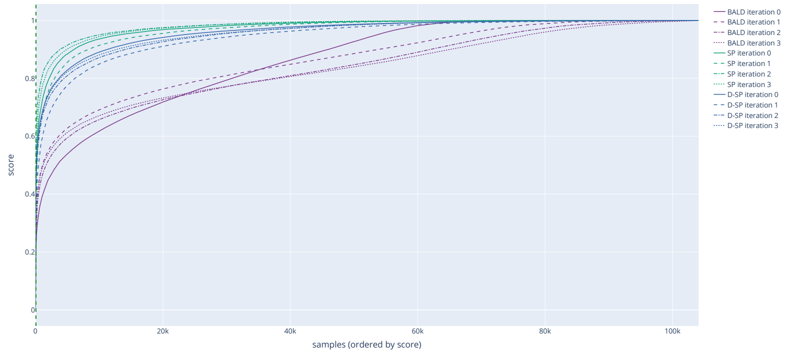
	BALD	SP	LS	RT	D-SP+RT	D-SP	RANDOM
BALD	200	95	88	86	88	87	96
SP	95	200	129	103	133	134	93
LS	88	129	200	99	116	111	85
RT	86	103	99	200	106	101	58
D-SP+RT	88	133	116	106	200	135	82
D-SP	87	134	111	101	135	200	88
RANDOM	96	93	85	58	82	88	200

(c) TechQA

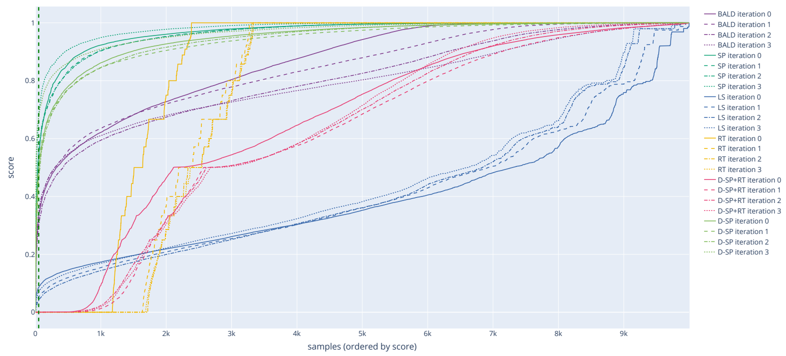
	BALD	SP	LS	RT	D-SP+RT	D-SP	RANDOM
BALD	200	39	44	42	54	49	42
SP	39	200	49	49	66	72	40
LS	44	49	200	37	48	46	41
RT	42	49	37	200	63	49	15
D-SP+RT	54	66	48	63	200	82	42
D-SP	49	72	46	49	82	200	36
RANDOM	42	40	41	15	42	36	200

(d) BioASQ

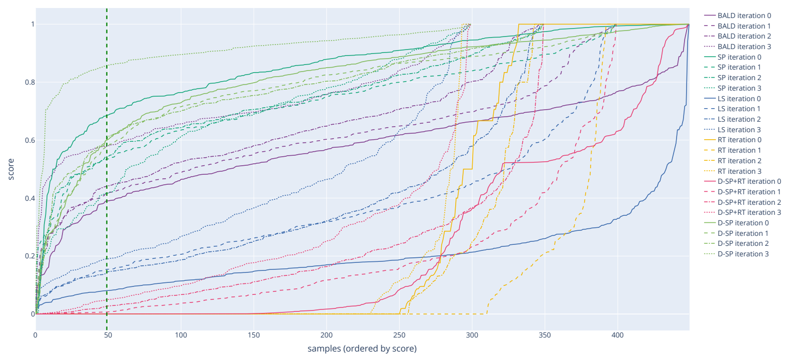
Table 6: Overlap for the domains: Although the overlap is quite high for TechQA, we explain this with the small amount of total samples available. In contrast we see a small overlap for large datasets like NQ.



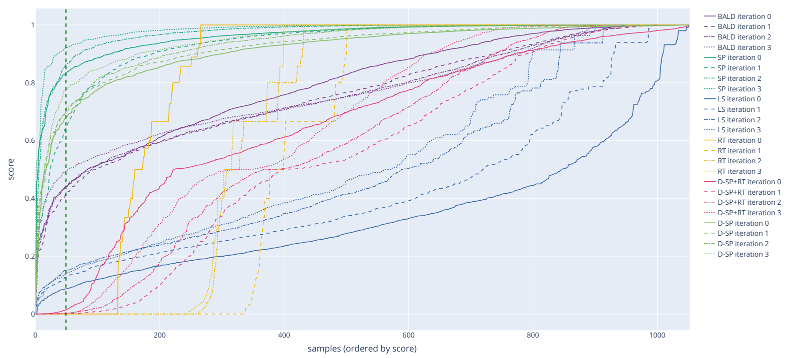
(a) Sample score distribution for NQ.



(b) Sample score distribution for NQ#10000.



(c) Sample score distribution for TechQA.



(d) Sample score distribution for BioASQ.

Figure 4: Sample score distributions for the datasets used in this work. Scores have been rescaled to $[0, 1]$.

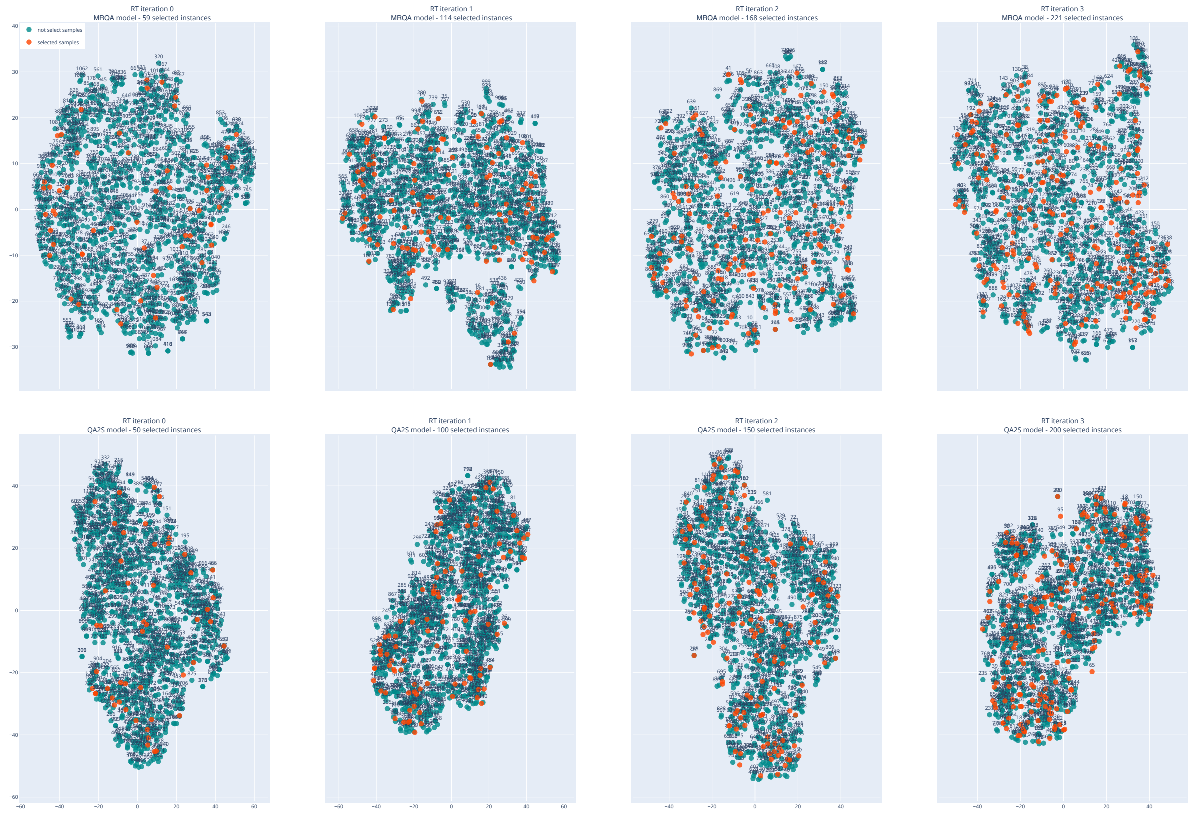


Figure 5: Visualization of sample encoding distribution on BioASQ with RT on both models, MRQA as well as data generation, via t-SNE of the last layer hidden states, sentence representation by averaging over tokens.