NTU ADL 2023 Fall Final Project
Domain Adaptive QA

Team 1
R11944074 汪宣甫
R12922104 蔡昀燁
R12922051 陳韋傑
R12922062 鄭雅勻
R12944062 李泓賢

Abstract

The rapid progress in Question Answering (QA) models has yielded impressive results on various datasets. However, a critical analysis reveals a notable limitation in their generalization capability, excelling in in-domain scenarios but struggling to handle out-domain variations. This project explores domain adaptation in QA to enhance model performance across diverse domains. For training, two backbone-model types, vanilla and context network-based, are applied, with the latter further paired with two embedding types: BERT-base and SimCSE. The experiments encompass not only in-domain and out-domain performance examination but also introduce a two-stage methodology involving pre-training on out-domain knowledge and fine-tuning on in-domain datasets. The results offer insights into trying to address domain variations in QA, contributing to the ongoing discourse on the development of robust and adaptable QA models for improved natural language understanding across diverse domains. Codes are available at https://github.com/ShampooWang/ADL-final_Domain-adaptive-QA.

● Introduction

The field of Question Answering (QA) has achieved remarkable advancements, with models showcasing impressive performances on various datasets. However, a closer inspection of these models reveals a notable deficiency in their generalization abilities. While excelling within their in-domain training datasets, these models exhibit limited adaptability when faced with variations in questions such as utilizing other datasets, i.e. out-domain scenarios. This discrepancy was highlighted in the previous work [1], and we just extract the table shown as below (*Table 1*).

		Evaluated on				
		SQuAD	TriviaQA	NQ	QuAC	NewsQA
Fine-tuned on	SQuAD	**75.6**	46.7	48.7	20.2	41.1
	TriviaQA	49.8	**58.7**	42.1	20.4	10.5
	NQ	53.5	46.3	**73.5**	21.6	24.7
	QuAC	39.4	33.1	33.8	**33.3**	13.8
	NewsQA	52.1	38.4	41.7	20.4	**60.1**

Table 1: F1 scores of each fine-tuned model evaluated on each test set

The table above illustrates a clear distinction between in-domain and out-domain scenarios, with the diagonal entries representing the former and the rest depicting the latter. It is evident from the F1 scores that the models' performance is significantly biased towards in-domain scenarios, signaling a challenge in generalization across similar tasks with different datasets. Motivated by these findings, our project delves into the domain adaptation aspect of QA, aiming to enhance model performance across varied domains. We employ two distinct backbone-model types: vanilla and context network-based. The latter is further coupled with two embedding types, respectively namely the widely used BERT-base and SimCSE [2].

To comprehensively evaluate the effectiveness of these configurations, our experiments not only examine in-domain and out-domain performance but also utilize a two-stage training methodology. This approach involves initially pre-training models on out-domain knowledge and subsequently fine-tuning them on in-domain datasets. Through a series of comparisons of experimental results, our objective is to find a strategy to solve the task for domain adaptive QA well. Our observations shed light on the challenges presented by domain variations in QA and provide valuable insights into the potential solutions.

Here is an illustrated overview of our pipeline (*Fig. 1* and *Fig. 2*):

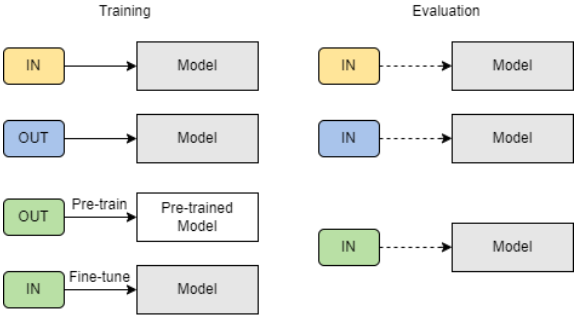

Figure 1: Training and evaluation

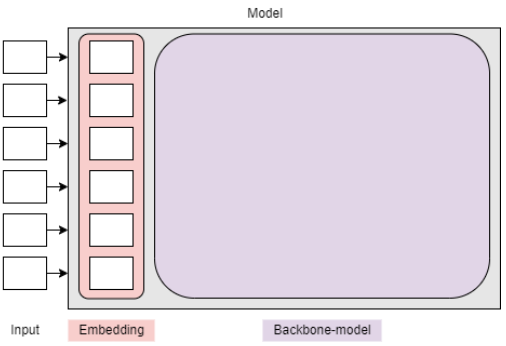

Figure 2: Model architecture

• Methods

As mentioned in the section of introduction, we will employ two backbone-model types with two embedding types, which finally comprises three distinct model types represpectively named “origin(base)”, “cls” and “sim”.

--model type	Backbone-model Type	Embedding Type
origin	Vanilla	BERT-based
cls	Context Network	BERT-based
sim	Context Network	SimCSE

Table 2: Definitions of all the model types

Vanilla

It is just a basic version of the model architecture, which is implemented without any specialized modifications or enhanced techniques. In details of implementation, we just utilize the model of *AutoModelForQuestionAnswering* with *bert-base-uncased*.

Context Network

By incorporating context, the network can improve its robustness and generalization capabilities of the overall model, which is particularly valuable in scenarios where the relationships between input features and the target output are context-dependent or subject to variations. As a result, the "context network" here typically refers to the mechanism within our neural network architecture that is designed to capture and leverage contextual information to enhance the model's ability to adapt to different domains. In details of implementation, we propose a new class called *ClsBertForQuestionAnswering* extended from *BertPretrainedModel*, but especially adding the mean value of the CLS token across the batch to each token as contextual information. The concept of proposed context network can be similarly depicted as below figure (*Fig. 3*) based on the previous work [3].

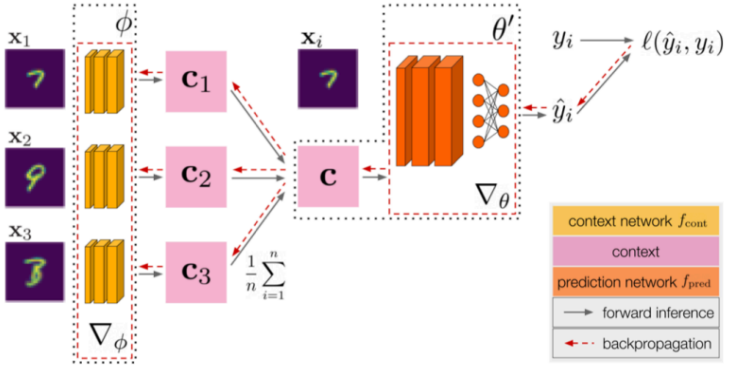

Figure 3. Context Network

BERT-based

BERT (Bidirectional Encoder Representations from Transformers) generates contextualized word embeddings by considering the bidirectional context of each word in a sentence, which is versatile and widely used for various natural language processing tasks.

SimCSE

Unlike BERT, which generates contextualized embeddings at the word level, SimCSE [2] directly targets sentence-level representations. It is an approach specifically designed for learning fixed-size embedding for entire sentences, and it utilizes the technique of contrastive learning to encourage similarity between embeddings of similar sentences and dissimilarity between embeddings of dissimilar sentences. The concept of proposed SimCSE can be depicted as below figure (*Fig. 4*) based on the previous work [2].

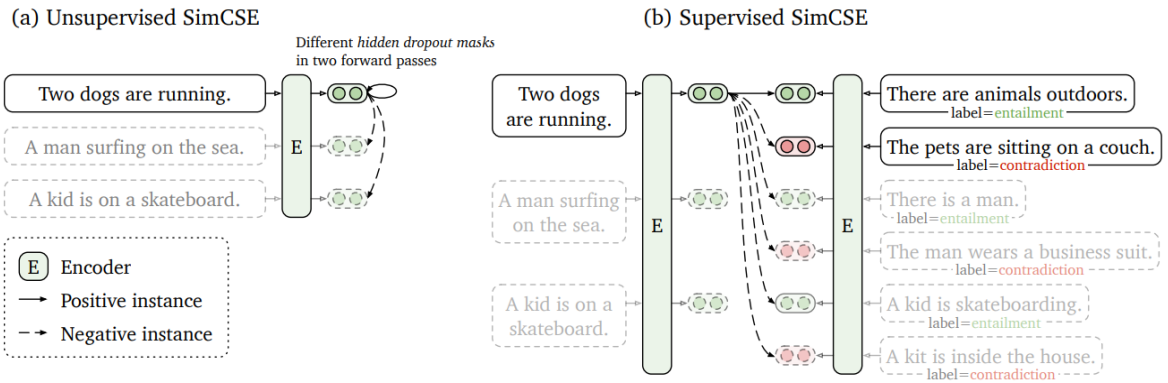

Figure 4. SimCSE

• Experiments & Results

We started our experiment on the base model (*--model type origin*). We conducted three types of experiments on 10 datasets, which were “in-domain”, “out-domain”, and “ODID”. Hence, there were totally 30 experiments on the base model. *Table 3* shows the 10 datasets we selected to use. As mentioned previously in *Fig 1*, “in-domain” refers to fine-tuning on the in-domain dataset itself, and then evaluating on the dataset itself. “Out-domain” refers to fine-tuning on the out-domain datasets (10-1=9 datasets), and then evaluating on the dataset itself. “ODID” is the most important one. It refers to pre-training on the out-domain datasets (10-1=9 datasets), then fine-tuning on the in-domain dataset itself, and finally, evaluating on the dataset itself.

In order to compare the three different experiments, we show the performance of each dataset with F1-score, based on the argument "*data_num*". Please be notified that "*data_num*" doesn't mean total training examples. However, it represents the number of examples extracted from each dataset. *Fig. 5* shows the details for you to understand more clearly. Besides, if the size of the dataset is smaller than the number of examples we want to extract for training, we will use the whole dataset.

comqa	duorc_p	duorc_s	hotpotqa	newsqa
nq	quac	squard	triviaqa	wikihop

Table 3: 10 datasets we use for the experiments

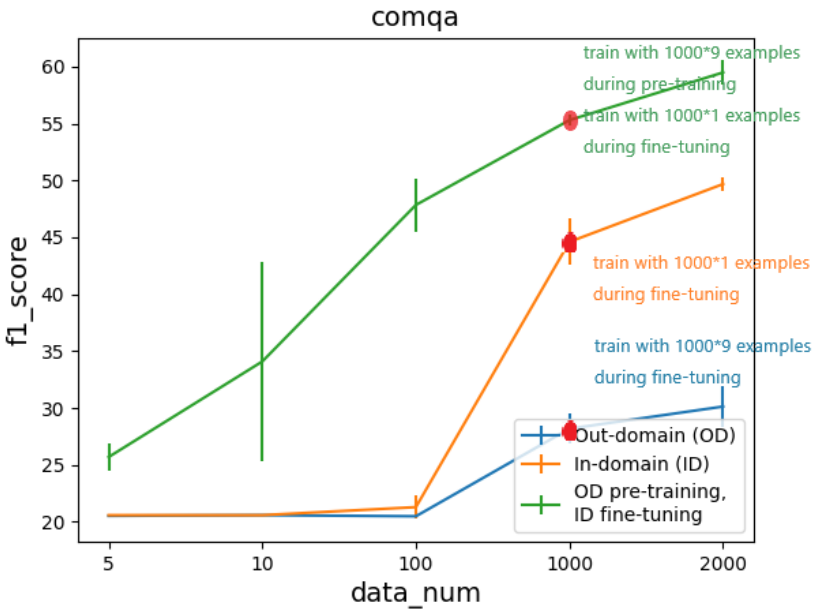

Figure 5: different total training examples of the same data_num

For each of 10 datasets, we tried five settings of *data_num*, which were 5, 10, 100, 1000, and 2000, respectively. Besides, each of 30 experiments share the same model settings, which are shown in *Table 4*.

num_train_epochs	2
learning_rate	3e-5
gradient_accumulation_steps	2
per_device_train_batch_size	12
max_seq_length	384
max_eval_samples	24

Table 4: model settings

Due to page limitation, here we only show the result of two datasets, while the whole result is shown in this link. *Fig. 6* shows the result of the "duorc_p" dataset. "ODID" (green line) always surpasses other 2 types of experiments given 5 different *data_num*. "Out-domain" (blue line) seems to yield quite satisfactory performance. "In-domain" (orange line) has the worst performance whatever the *data_num* is. Due to our definition of *data_num*, you may regard this performance as an unreasonable result. We will talk about this later in *Discussion*. *Fig. 7* indicates the result of the "comqa" dataset. "ODID" always has the best performance given 5 different *data_num*, which is the same as the "duorc_p" dataset. However, this time, "in-domain" makes a greater performance than "out-domain" when *data_num* is larger than 100. This yields a reasonable result since the in-domain knowledge provides the model with more informative resources than the out-domain knowledge.

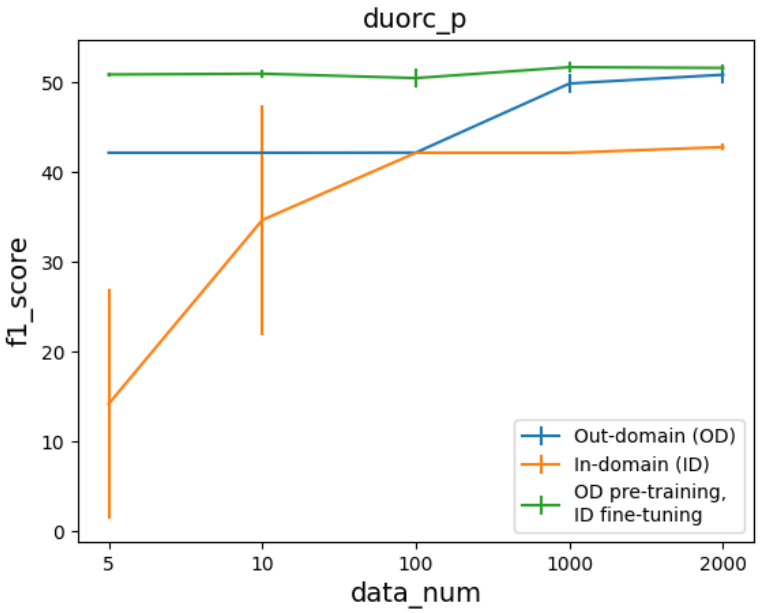

Figure 6: results of duorc_p dataset

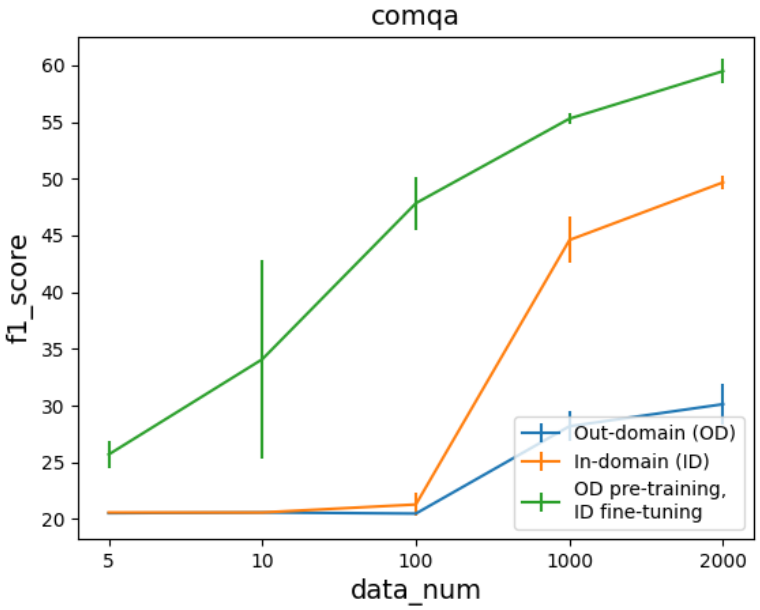

Figure 7: results of comqa dataset

After conducting 30 experiments on the base model (*--model type origin*), we also tried the other two methods as previously mentioned on the "comqa" dataset. That is, "cls" and "sim". We expected that we would get a better performance because of the improved embedding space. However, we got an unanticipated result on three types of experiments. Take "ODID" for example, *Fig. 8* and *Table 5* show the result of it. We can barely conclude that either "cls" or "sim" leads to a better performance.

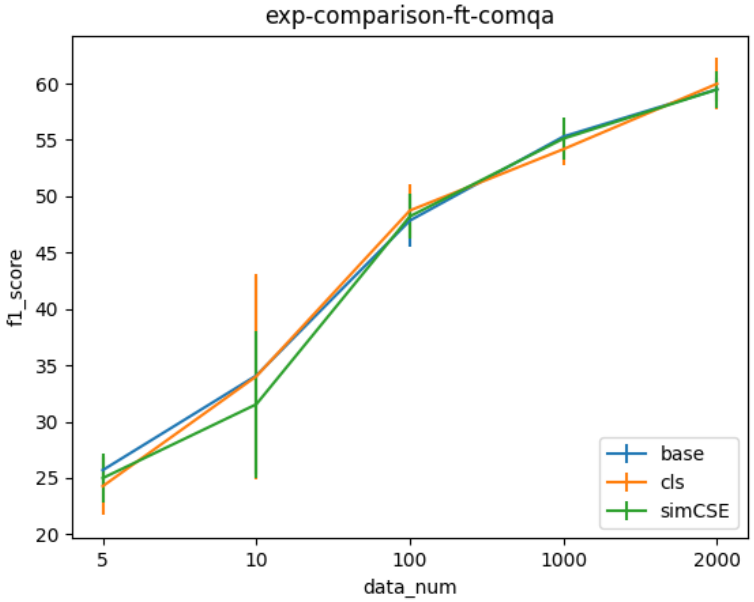

Figure 8: ODID performance on three different methods on the "comqa" dataset

	data_num=5	10	100	1000	2000
base	**25.694±1.250**	**34.089±8.732**	47.845±2.363	**55.301±0.437**	59.490±1.048
cls	24.268±2.583	34.021±9.125	**48.737±2.337**	54.183±2.337	**59.987±2.267**
sim	24.986±2.158	31.538±6.509	48.221±2.005	55.119±2.005	59.482±1.619

Table 5: ODID performance details

• Discussion & Conclusion

There are two issues we want to discuss. First, according to the result of the "duorc_p" dataset in *Fig. 6*, out-domain (blue line) has a better performance than in-doman (orange line), especially in the few-shot setting (5~100 data_num). Let's not focus on the definition of the *data_num* now. We find that out_domain experiment tends to predict "no answer" in the few-shot setting. Then, we take notice of the ground truth of the evaluation set of the "duorc_p" dataset. We discover that there are lots of "no answers" in the ground truth set. The distribution of the evaluation set is shown in *Fig. 9*. We conclude that this is the reason why we get this consequence.

Second, let's take a look at *Fig. 10*. We want to emphasize that, if you only have some out-domain data, it's pretty difficult to get the same performance as the identical number of in-domain data. That is, you should possess a huge amount of out-domain data to guarantee the same performance. *Fig. 9* illustrates this concept: In order to outperform the in-domain(orange line) training with 300 examples, you should train your model with more than 9000 out-domain examples. However, we don't focus on the exact value of proportion, i.e., 300/9000. Instead, we want to conclude that one can train a model with abundant out-domain data while possessing insufficient in-domain data, due to the difficulty of getting enough in-domain data.

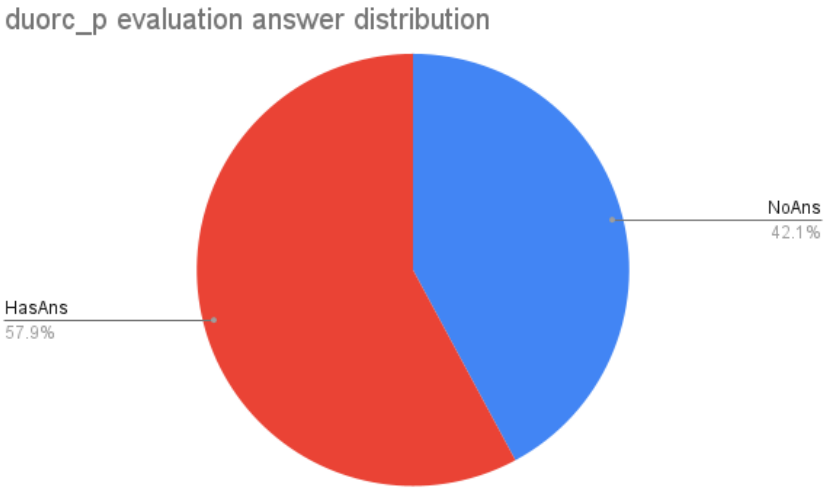

Figure 9: distribution of the evaluation set of duorc_p dataset

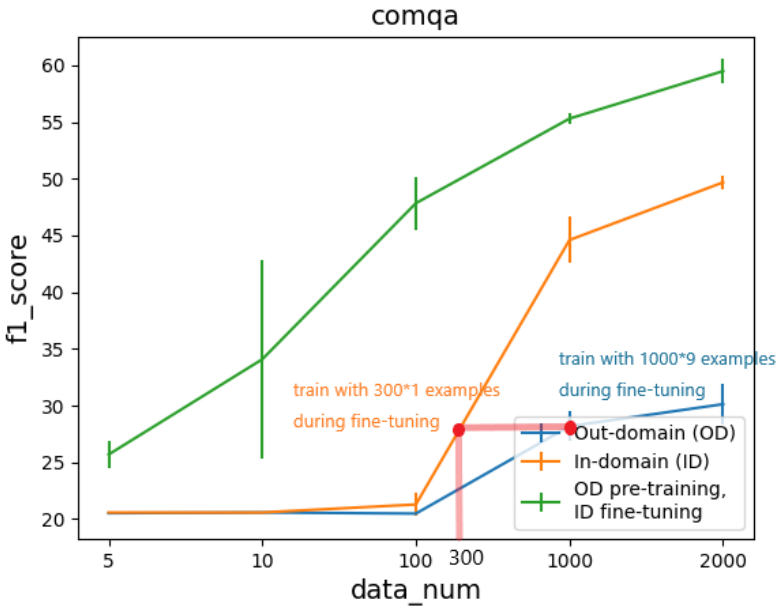

Figure 10: an example of how to get the same performance

- # References

[1] Priyanka Sen, Amir Saffari. What do Models Learn from Question Answering Datasets? In *EMNLP 2020.*

[2] Tianyu Gao, Xingcheng Yao, Danqi Chen. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *EMNLP 2021.*

[3] Marvin Zhang, Henrik Marklund, Nikita Dhawan, Abhishek Gupta, Sergey Levine, Chelsea Finn. Adaptive Risk Minimization: Learning to Adapt to Domain Shift. In *NeurIPS 2021.*