Continual Domain Adaptation for Machine Reading Comprehension

Lixin Su, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Yanyan Lan, and Xueqi Cheng
CAS Key Lab of Network Data Science and Technology, Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, China
University of Chinese Academy of Sciences, Beijing, China
sulixinict@gmail.com, {guojiafeng, zhangruqing, fanyixing, lanyanyan, cxq}@ict.ac.cn

ABSTRACT
Machine reading comprehension (MRC) has become a core component in a variety of natural language processing (NLP) applications such as question answering and dialogue systems. It becomes a practical challenge that an MRC model needs to learn in non-stationary environments, in which the underlying data distribution changes over time. A typical scenario is the domain drift, i.e., different domains of data come one after another, where the MRC model is required to adapt to the new domain while maintaining previously learned ability. To tackle such a challenge, in this work, we introduce the Continual Domain Adaptation (CDA) task for MRC. So far as we know, this is the first study on the continual learning perspective of MRC. We build two benchmark datasets for the CDA task, by re-organizing existing MRC collections into different domains with respect to context type and question type, respectively. We then analyze and observe the catastrophic forgetting (CF) phenomenon of MRC under the CDA setting. To tackle the CDA task, we propose several BERT-based continual learning MRC models using either regularization-based methodology or dynamic-architecture paradigm. We analyze the performance of different continual learning MRC models under the CDA task and show that the proposed dynamic-architecture based model achieves the best performance.

CCS CONCEPTS
• Information systems → Question answering.

KEYWORDS
Machine Reading Comprehension, Continual Domain Adaptation, Dynamic Network

1 INTRODUCTION
Machine Reading Comprehension (MRC) aims to read and understand unstructured texts and then answer questions about it, which has become a core component in a variety of natural language processing (NLP) applications [5]. For example, in retrieval-based question answering systems, e.g., Web QA [6], MRC plays a vital role which attempts to comprehend the retrieved documents to extract a concise answer for a given question. In dialogue systems, e.g., conversational information assistant [30], MRC is often employed as a basic component to build professional response skills over some target information resources such as medical or legislative corpus.

As a component of real-world applications, it becomes a practical challenge that an MRC model needs to learn in non-stationary environments. That is the underlying data distribution where a model is applied changes over time. A typical case is called domain drift [24], where different domains of data come one after another. For example, a question answering system may gradually enlarge its retrieved repository from news articles to social media, or from factoid information to non-factoid information. A dialog system, e.g., a medical information assistant, may consistently expand its skill to provide information about more and more types of diseases.

These practical issues require the underlying MRC model to adapt to the new domain while maintaining previously learned knowledge. Note that in practical scenarios, an early domain of data may not be available in a later phase due to some privacy or storage issues.

To study such practical challenge, in this work, we introduce the Continual Domain Adaptation (CDA) task for MRC. In this CDA task as shown in Figure 1, training data from different domains arrives in a sequential manner, and an MRC model is required to continually learn over each domain. Evaluation of CDA is to test the overall performance on all the domains that the MRC model
has encountered. As we knew, there have been a large variety of MRC tasks [12] proposed in the literature. However, all those tasks assume a stationary learning scenario, i.e., a fixed data distribution. So far as we know, this is the first task about the continual learning ability of MRC models.

To facilitate the study of the CDA problem, we build two benchmark CDA datasets for MRC, namely CDA-C and CDA-Q, by re-organizing existing MRC collections into different domains with respect to context type and question type, respectively. For the CDA-C dataset, we make use of five MRC collections including SQuAD [31], NewsQA [39], TweetQA [43], NarrativeQA [21], and DuoRC [34]. We obtain a 5-domain dataset to mimic the drift of the context distribution, from wiki passages, news articles, movie scripts, book summaries to tweets. For the CDA-Q dataset, we make use of the NaturalQA [22] collection and split it into 8 domains according to the question types such as what, why, and how. Both datasets would be publicly available for the research community. Based on these two CDA datasets, we conduct some empirical analysis over the existing MRC models to check whether the MRC model suffers from the domain drift problem. We employ a state-of-the-art MRC model, i.e., BERTQA [10], for the analysis. We observe the very obvious forgetting phenomenon of the BERTQA model under the CDA setting 4 similar to [26], i.e., a general tendency to forget past knowledge as it learns over new domains.

To tackle the CDA task for MRC, we propose to equip the existing state-of-the-art MRC models with the continual learning ability. Continual learning is an exciting direction in the machine learning community, which has been actively studied in the computer vision (CV) area in recent years [25, 26, 28, 35] but received less attention in NLP. The exiting continual learning techniques could be categorized into two major branches, i.e., regularization-based methods and dynamic-architecture methods. Regularization-based methods [20] identify important weights for previous domains and heavily penalize their deviation while learning new domains with some type of regularization. Dynamic-architecture methods [33] append new components or eliminate learned weights in the network when learning with continually arrived domains.

We borrow the idea from the CV area and propose several BERT-based continual learning MRC models using either regularization-based methodology or dynamic-architecture paradigm. Based on the regularization-based methodology, we propose RegBERTQA which adds a penalty to the learning objective of the BERTQA model which restricts the change of parameters to prevent forgetting. Based on the dynamic-architecture paradigm, we introduce a novel ProgBERTQA model for MRC, which enlarges its model capacity progressively when a new domain arrives. Specifically, the ProgBERTQA model leverages the BERT model as a backbone structure and adds an adapter component for each incoming domain. The adapter is learned for the current domain while the backbone parameters which are domain-shared are kept unchanged all the time. We studied two types of adapter architectures and two ways to insert the adapter into the BERT model. We conduct extensive experiments to analyze the performance of different continual learning MRC models under the CDA task. Based on the empirical results, we find that the ProgBERTQA, which adopts the dynamic-architecture paradigm, achieves the best performance on the CDA task for MRC.

The major contributions of this paper include:
1. We introduce the CDA task, which is the first task focusing on the continual learning ability of MRC models, and show that existing MRC models do suffer from the CF phenomenon under the CDA setting.
2. We build two publicly available benchmark datasets that could facilitate the study of the CDA problem of MRC in the future.
3. We propose and investigate several BERT-based continual learning MRC models, which could combat the CF phenomenon while learning over new domains continually.

2 RELATED WORK

In this section, we first briefly review two lines of related work, i.e., machine reading comprehension and continual learning. Then, we introduce various datasets for machine reading comprehension research in detail.

2.1 Machine Reading Comprehension

Teaching machines to read and comprehend is a fundamentally interesting and challenging problem. MRC plays a vital role in a question answering system for comprehending retrieved passages to achieve a concise answer [6]. Existing literature on MRC models has focused on the standard supervised setting, where the training data and the test data are assumed to sample from the same distribution. BiDAF [36] takes RNN for representing the context and the question. Consequently, bidirectional attention flow mechanism is applied to obtain a query-aware context representation. This context representation is further used to predict the start position and the end position of the answer. FusionNet [16] also applies RNN for text representation and proposes a fully-aware multi-level attention mechanism to capture hierarchical information of each word. QANet [44] applies the structure consisting of convolution and self-attention for fast training. Convolution layer captures local interactions, and self-attention mechanism perceives global interactions. BERT [10] is a self-supervised approach for pre-training a deep transformer encoder [40]. BERTQA is a fine-tuned BERT for the MRC task and achieves state-of-the-art performance. In this paper, different from existing MRC models designed for the stationary dataset, our goal is to propose an MRC model that can handle domain drift to achieve continual domain adaptation.

2.2 Continual Learning

Continual learning aims to sequentially learn a series of tasks. Existing methods for continual learning can be divided into two main categories: network regularization and dynamic architecture. In the following, we give a brief review of typical methods in two categories.

2.2.1 Network regularization. Regularization-based approaches suggest using a regularizer that prevents the parameters from drastic changes in their values yet still enables them to converge to a good solution for the new task [4, 11, 20, 25, 25]. To keep the learned knowledge, various penalties are added to the change of weights. EWC [19] uses Fisher’s information to evaluate the importance of weights for old tasks, and updates weights according to the
degree of importance. Based on similar ideas, the method in [45] calculates the importance based on the learning trajectory. Online EWC [35] and EWC++ [4] improve the efficiency issues of EWC. The approach builds an attention map and requires that the attention region of the previous and concurrent models are consistent. There are related methods [37] using additional models to remember previous data distribution to regularize the learning process. Generative Replay [37] introduces GANs to lifelong learning, which uses a generator to sample fake data. New tasks can be trained with these generated data.

These regularization-based methods can be applied to different model structure, and we adapt representative methods EWC to MRC model for CDA.

2.2.2 Dynamic Architecture. Dynamic-architecture approaches [26] dynamically increase model capacity during training. ProgressiveNet [33] expands the architecture for new tasks and keeps previous knowledge by preserving the previous weights. LwF [26] divides the model layers into two parts, i.e., shared and task-specific layers. The former are shared by tasks, and the later are expanded for new tasks. DAN [9] extends the architecture for each new task, and the newly added layer is a sparse linear combination of the original filters in the corresponding layer of a base model. Asghar et al. [2] augment parameterized memory for sequentially arrived tasks in RNN structure for text classification task.

These dynamic methods cannot be applied to MRC model, as they are design for CV models or RNN structure. Inspired by these dynamic architectures, we aims to design a dynamic MRC model based pre-trained transformer for CDA task.

2.3 MRC Datasets
There are many released MRC datasets for developing and evaluating MRC models, which can be generally categorized into basic datasets and derived datasets.

2.3.1 Basic Datasets. MRC datasets can be categorized into three types with respect to questions. Cloze-style MRC datasets [14] aim to teach machines to read the passage and fill the blank in questions. Multiple-choice MRC datasets [23] give the context, the question, and several candidate answers, then asks models to choose the correct answer. Extractive MRC [31] aims to locate text span from the context given the context and the question. Extractive MRC has attracted extensive attentions, since it can be used in real question answering system and question answering system [30]. In this paper, we focus on extractive MRC and we ignore “extractive” for brevity. We then describe some details of MRC datasets. MRC datasets such as SQuAD, NarrativeQA, and HotpotQA contain question-passage-answer triple. Questions are written by annotators after they read context passages, and answers are short text spans in the passages. SQuAD [31] are constructed based on Wikipedia passages. NewsQA [39] are constructed based on CNN news. NarrativeQA [21] use the passages from fictional story books, which exposes additional challenge for longer passages. NaturalQA [22] is a more natural dataset, as its questions consist of real anonymized, aggregated queries issued to the Google search engine and its context passages are Wikipedia page from the top 5 search results. NaturalQA effectively eliminates the plagiarism from passage to question as question is collected before contexts. TweetQA [43] presents the first large-scale dataset for QA over social media data. It gathers tweets as context passages and human annotators write questions and answers upon these tweets. DuoRC [34] is constructed from a collection of movie plot pairs where each pair in the collection reflects two versions of the same movie written by different authors. questions are generated based on one version and answers are synthesized from the other version.

2.3.2 Derived Datasets. Given extensive MRC datasets, some works reused those datasets to comprehensively test the high-level ability of models. GLUE [41] collected a series of NLU tasks including question answering, sentiment analysis, and textual entailment. GLUE is a multitask benchmark in NLP, designed to encourage models that share general linguistic knowledge across tasks. Wang et al. [42] grouped SQuAD, NewsQA, and MS MARCO to study the unsupervised domain adaptation for MRC. They used a full MRC dataset as the source domain and take only unlabeled passages in another dataset as the target domain. With this derived dataset, they test the model capability of unsupervised domain adaptation. Fisch et al. [12] presented Machine Reading for Question Answering (MRQA) 2019 shared task, which tested extractive MRC models on their ability to generalize to data distributions different from the training distribution. They unified 18 distinct question answering datasets into the uniform format. Among them, six datasets were made available for training, six datasets were made available for development, and the final six were hidden for final evaluation.

Different from existing derived datasets, we aim to construct a continual domain adaptation setting. We split existing datasets into different domains, and each domain is available to the MRC model sequentially.

3 CONTINUOUS DOMAIN ADAPTATION FOR MRC
In this section, we introduce the Continual Domain Adaptation (CDA) task for MRC and describe the benchmark CDA datasets for MRC in detail.

3.1 Task Formulation
In the CDA task for MRC, we assume that there exists T MRC datasets \( \mathcal{D}_1, \ldots, \mathcal{D}_T \) from T different domains arriving in a sequential manner. We denote the dataset \( \mathcal{D}_t \) from domain \( t \) as,

\[
\mathcal{D}_t = \{c^t_i, q^t_i, a^t_i\}_{i=1}^{N_t}
\]

(1)

where \( c^t_i, q^t_i \) and \( a^t_i \) denotes the \( i \)-th context, question and answer among \( N_t \) samples. In the CDA setting, an MRC model is required to continually learn over each domain dataset \( \mathcal{D}_t \), and evaluated on all seen domains. Each time, the MRC model can only access current domain and cannot access previous training domains due to the privacy or storage issues.

The training objective at the arrival of domain \( t \) can be defined as follows,

\[
\min_{\Theta_t} \mathcal{L}(\Theta_t; \Theta_{t-1}, \mathcal{D}_t) + \lambda R(\Theta_t),
\]

(2)

where \( \Theta_t \) denotes the set of parameters for the MRC model at domain \( t \), and \( R(\cdot) \) is a regularization term on the model parameters.
In order to study and evaluate the CDA model for MRC, we build two benchmark datasets, i.e., CDA-C and CDA-Q. Datasets are constructed based on several existing MRC collections by splitting them into different domains with respect to context type and question type respectively. We now describe the detail of the CDA-C and CDA-Q dataset as follows.

### 3.2.1 CDA Dataset for Context Drift
To mimic the drift of the context distribution in the CDA task, we construct a new benchmark dataset called CDA-C. We take five MRC collections including SQuAD from Wiki passages [31], NewsQA from CNN news articles [39], TweetQA from tweets [43], NarrativeQA from book summaries [21], and DuoRC from movie scripts [34] as our whole MRC datasets, since (1) These collections are publicly available and the size of each dataset is big enough for training deep neural models; (2) The contexts in these collections are different from each other and it is reasonable to distinguish one domain from another domain. Thus, we can obtain a 5-domain datasets to mimic the drift of the context distribution. Specifically, for saving computation cost, we randomly sample 10,000 <question, context, answer> triples from each MRC datset as the training set, and the original test set is used for testing. Table 1 shows the overall statistics of our CDA-C benchmark dataset.

### 3.2.2 CDA Dataset for Question Drift.
In order to mimic the drift of the question distribution in CDA task, we build a new benchmark dataset CDA-Q. Here, we take the NaturalQA [22] dataset as our source data. The reason is that NaturalQA contains various types of questions, and the amount of questions are much larger than other datasets [31]. Specifically, we first select 7 question words (such as "what", "how" and "why") to represent the question types. Then, to identify the question type for each question, we locate the question word in the question if it can be found in the first three tokens, or the last word otherwise. Note a question would be defined as "other" type if there exists no question words in it. In this way, we obtain a 8-domain dataset to mimic the drift of the question distribution. Table 2 shows the overall statistics of our CDA-Q benchmark dataset.

### 4 ANALYSIS OF BERTQA MODEL ON CDA
In this section, based on our constructed CDA-C and CDA-Q benchmark dataset, we conduct some empirical analysis over the existing MRC model to investigate whether the MRC model suffers from the domain drift problem, i.e., a sequence of continuously evolving domains, which has been identified in many computer vision methods [19].

Great efforts have been made to develop MRC models for better effectiveness. A significant milestone is that the pre-trained BERTQA model [10] has achieved state-of-the-art performance on several widely used MRC datasets [8, 32], and even exceeded the performance of human annotators on the SQuAD dataset [31]. Hence, we employ this representative BERTQA model for the analysis, and the model detail is described in section 5.1.

To evaluate the performance of BERTQA on the CDA datasets, we use pre-trained BERT trained on English Wikipedia, and we finetune it when each domain arrives. The BERTQA parameters learned from previously seen domains will be continually fine-tuned to the new domain during training. We report F1 performance on each domain in the test CDA-C and CDA-Q dataset.

Table 3 shows the evaluation results on test CDA-C dataset. We can observe that BERTQA performance on all earlier domains significantly drops as new domains arrive. For example, the drop F1 in the wiki domain at the last step over that at the first step is about 12.2. The main reason might be that BERTQA forgets how to solve old domains after being exposed to a new one due to interference caused by parameter updates. Table 4 shows the evaluation results on test CDA-Q dataset. We can see consistency performance drop.

---

### Table 1: Dataset statistic of CDA-C.

| datasets | wiki | news | scripts | book | tweet |
|----------|------|------|---------|------|-------|
| #train   | 10,000 | 10,000 | 10,000 | 10,000 | 10,000 |
| #test    | 10,507 | 4,212 | 9,344 | 9,227 | 4212 |
| #q_words | 10.2  | 6.5  | 7.4   | 8.4  | 6.5  |
| #a_words | 3.0   | 3.9  | 1.6   | 1.8  | 3.9  |
| #c_words | 124.0 | 492.6 | 658.2 | 580.1 | 492.6 |

### Table 2: Dataset statistic of CDA-Q.

For the evaluation of CDA, we employ the overall performance on all seen domains that the MRC model has been trained on,

\[
\sum_{t} \sum_{q_{t}, c_{t} \in D_{t}} g(a_{t}, f(q_{t}, c_{t}; \Theta_{t})),
\]

where \(f(\cdot)\) denotes the learned MRC model and \(g(\cdot)\) denotes the widely used evaluation metric for MRC, e.g., EM and F1. The detail about evaluation will be described in section 6.2.

### Table 3: F1 of BERTQA on CDA-C. Domain sequence is wiki>news>scripts>book>tweet.

| datasets | wiki | news | scripts | book | tweet |
|----------|------|------|---------|------|-------|
| #train   | 16809 | 3052 | 12136 | 19470 | 5791 |
| #test    | 2221  | 383  | 1501  | 2256  | 653  |
| #q_words | 9.3   | 10.6 | 8.7   | 8.9   | 9.3  |
| #a_words | 6.3   | 7.2  | 3.5   | 3.6   | 15.7 |
| #c_words | 153.7 | 168.5| 126.4 | 158.6 | 166.6|

### Table 4: F1 of BERTQA on CDA-Q.

For the evaluation of CDA, we employ the overall performance on all seen domains that the MRC model has been trained on,

\[
\sum_{t} \sum_{q_{t}, c_{t} \in D_{t}} g(a_{t}, f(q_{t}, c_{t}; \Theta_{t})),
\]

where \(f(\cdot)\) denotes the learned MRC model and \(g(\cdot)\) denotes the widely used evaluation metric for MRC, e.g., EM and F1. The detail about evaluation will be described in section 6.2.

### Table 3: F1 of BERTQA on CDA-C. Domain sequence is wiki>news>scripts>book>tweet.

| datasets | wiki | news | scripts | book | tweet |
|----------|------|------|---------|------|-------|
| #train   | 80.01 | -    | -       | -    | -     |
| #test    | 75.63(-5.5%) | 58.5 | -    | -    | -     |
| #q_words | 71.08(-11.2%) | 44.35(-24.2%) | 67.33 | - | - |
| #a_words | 69.97(-12.5%) | 44.25(-24.4%) | 67.73(+0.6%) | 61.21 | - |
| #c_words | 67.9(-15.1%) | 44.69(-23.6%) | 67.8076(+0.7%) | 58.51(-4.4%) | 85.13 |
The BERTQA model is formed by incorporating BERT [10] with similarity between transformed representations. Concretely, the transitions, where the attention weights are derived by the dot-product information from different representation subspaces at different positions. Multi-head attention allows the model to jointly attend to global information through multi-head attention (MH) and a linear network and layer normalization. We denote the process as SA(.):

\[
\text{SA}(\cdot) = \text{MH}(\cdot) + \text{FFN}(\cdot)
\]

The Self-Attention sub-layer aims to capture global information through multi-head attention (MH) and a linear layer. Multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions, where the attention weights are derived by the dot-product similarity between transformed representations. Concretely, the \(i\)-th attention mechanism “head” is:

\[
\text{Attention}_i(h_j) = \sum_t \text{softmax} \left( \frac{W^q_i h_j \cdot W^k_i h_t}{\sqrt{d/|n|}} \right) W^v_i h_t
\]

where \(h_j\) is a \(d\)-dimensional hidden vector for a particular sequence token, \(W^q_i, W^k_i\) and \(W^v_i\) are learned matrices of size \(d/n \times d\), and thus each “head” projects down to a smaller subspace of size \(d/n\). Finally the outputs of the \(n\) attentions heads are concatenated together and passed to a linear transformation:

\[
\text{MH}(h) = \text{W}^\omega [\text{Attention}_1(h), \ldots, \text{Attention}_n(h)]
\]

with \(\text{W}^\omega\) is a \(d \times d\) matrix. The outputs are further passed to residual network and layer normalization. We denotes the process as \(\text{SA}(\cdot)\):

\[
\text{SA}(h) = \text{LN}(h + \text{MH}(h))
\]

where \(\text{LN}(\cdot)\) is layer normalization [3], requiring \(2d\) parameters.

5.1.2 Feed Forward Network. The outputs of \(\text{SA}(\cdot)\) is passed to two-layer feed forward network (FFN). FFN is shown as

\[
\text{FFN}(h) = W_2 f(W_1 h + b_1) + b_2
\]

where matrix \(W_1\) has size \(d_{ff} \times d\) and \(W_2\) has size \(d \times d_{ff}\), \(f(\cdot)\) is an activation function, so overall we required \(2dd_{ff}\) parameters from the FFN layer.

Putting this together, a BERT layer, which we denote as \(\text{BL}(\cdot)\), is a layer-norm applied to the output of FFN layer, with a residual connection.

\[
\text{BL}(h) = \text{LN}(\text{FFN}(\text{SA}(h)) + \text{SA}(h))
\]

We have \(4d^2 + 2dd_{ff}\) total parameters from a BERT layer.

5.2 RegBERTQA

Based on the regularization-based methodology in continual learning [20], we propose the RegBERTQA model which adapts EWC [20] method to BERTQA. The key idea is that it is reasonable to prevent forgetting by restricting the change of parameters to keep the network parameters close to the learned parameters of the old domain.

Specifically, RegBERTQA updates model parameters based on the importance evaluated in previously seen domains. We denote parameters of BERT at \(t\) domain as \(\Theta_t\). When \(t-\text{th}\) domain arrives, we fine-tune BERTQA and add a penalty, weighted distance between old parameters \(\Theta_{t-1}\) and \(\Theta_t\), denoted as \(R(\Theta_t) = \sum_i f_i(\Theta_{t-1} - \Theta_t)^2\)
The model size of ProgBERTQA is 

\[ N_{\text{BERT}} + T + N_{\text{adapter}}, \]

where \( N_{\text{BERT}} \) is the amount of BERT parameters, \( N_{\text{adapter}} \) is the amount of adapter parameter, \( T \) is the number of domains. As the model size grows linearly with the number of domains, how to design an effective adapter which could achieve better performance with limited parameters is the core problem in ProgBERTQA. We now describe different ways of adapter insertion and adapter structure.

5.3.2 The Adapter Insertion. In this section, we discuss where to insert the adapter into the BERTQA model.

Adding the adapter to each BERT layer is more parameter efficient than that to the topmost layer [38], i.e., achieving more effective capacity with the same number of parameters. Based on the previous introduction, each BERT layer contains a self-attention and a feed forward layer. We then add the adapter to these two components. As depicted in Figure 2, we explore two position to insert adapter, namely, inside and aside. In inside fashion, as shown in Figure 2 inside, we put an adapter following the self attention component, another following the feed forward layer.

Inside fashion use adapter to reform the outputs of multi-head attention and feed forward layer respectively.

\[ SA(h) = LN(Adapter(MH(h)) + h), \]

\[ BL(h) = LN(Adapter(FFN(SA(h))) + SA(h)), \]

where \( Adapter(\cdot) \) denotes the adapter layer. In inside fashion, as shown in Figure 2 aside, we add two adapters parallel with multi-head attention and feed forward layer respectively.

\[ SA(h) = LN(MH(h) + Adapter(h) + h), \]

\[ BL(h) = LN(FFN(SA(h)) + Adapter(SA(h)) + SA(h)). \]

Inside fashion use adapter outputs as addend to the outputs of multi-head attention and feed forward layer. Aside fashion use adapter outputs to efficiently enlarge its capacity, thus we investigate these two structures in CDA for MRC. In the next section, we describe these two structures in detail.

5.3.3 The Adapter Structure. In this section, we discuss how to design the adapter structure.

There are many choices on the adapter structure and we introduce two structures based on the attention and feed-forward layer respectively. The reason is that they have achieved a more effective model capacity with same number of parameters on multi-task learning [38] and transfer learning [15]. ProgBERTQA aims to efficiently enlarge its capacity, thus we investigate these two structures in CDA for MRC. In the next section, we describe these two structures in detail.

As depicted in Figure 3, both these two structures are surrounded by two projection layers. The first projection layer transforms the origin space of \( d \) dimension to the space of \( d_i \) dimension to save the budget of parameters. The output of the first projection is passed to a transformation layer. Finally, another projection layer transforms
the vector to the space of $d$ dimension. These two structures are only different in the transformation layer. We list them as follows.

- Inspire by [38], a multi-head attention layer is utilized as the transformation layer. The intuition behind this AL is that different domain needs different interaction between the token representations.
- Inspire by [15], a non-linear activation function gelu [13] is used as transformation layer. The whole adapter is a bottleneck architecture.

We denote these two adapter structures as Projected Attention Layer (PAL) and BottleNeck (BN) respectively.

We can see that the number of parameters in PAL is $3d_s d_s + 2d_s d$ and the number of parameters in BN is $2d_s d$. In order to fairly compare these two structure we keep the number of parameters in each structure the same. For example when $d_s$ in BN is 256, then the $d_s$ in PAL is set to 192.

By combining these two ways of adapter insertion (i.e., inside and aside) and two adapter structures (i.e., PAL and BN), we obtain four types of ProgBERTQA models denoted as ProgBERTQA$^{I-PAL}$, ProgBERTQA$^{I-BN}$, ProgBERTQA$^{A-PAL}$, ProgBERTQA$^{A-BN}$.

5.4 Model Training and Testing

In the training phase, we sequentially training the RegBERTQA and ProgBERTQA on each domain. For RegBERTQA, we update all the parameters. For ProgBERTQA, we update only the adapter and the MRC output layer. We do not update the shared BERT parameter and keep them unchanged. The training loss of ProgBERTQA is the same as that of BERTQA [10].

In the testing phase, we test the model on all its seen domain. In ProgBERT model, domain label is needed, which indicates which adapter is activated to predict the results.

6 EXPERIMENTS

In this section, we conduct experiments to verify the effectiveness of our proposed models.

6.1 Experimental Settings

We implement our models in PyTorch [29] based on Transformers library.

We optimize the model using Adam [18] with warmup technique, where the learning rate is increased over the first 10% of batches and then decayed linearly to zero. All runs are trained on Tesla V100 GPU with batch size 16. We train the model for each domain for 3 epochs and use the result from the last epoch.

The learning rate is tuned amongst $\{2e^{-5}, 5e^{-5}\}$. We use the base-uncased version of BERT and other related pre-trained BERT-related models can also be tried, such as RoBERTa [27], spanBERT [17].

The maximum sequence length of BERT is set to 512. Datasets and codes are available at https://github.com/lixinsu/CDA-CIKM2020.

6.2 Evaluation Metrics

Inspired by [2], we report the answer accuracy of each domain. We also show the overall performance of all domains by the sum of the accuracy in each domain.

6.3 Models

6.3.1 Our Models. We presents our proposed methods based on regularization and as follows:

- RegBERTQA: Add a regularization term to prevent forgetting.
- ProgBERTQA$^{I-PAL}$: Inserting projected attention layer inside each of the BERT layers.
- ProgBERTQA$^{I-BN}$: Inserting bottleneck layer inside each of the BERT layers.
- ProgBERTQA$^{A-PAL}$: Inserting projected attention layer inside each of the BERT layers.
- ProgBERTQA$^{A-BN}$: Inserting bottleneck layer inside each of the BERT layers.

6.3.2 Baselines. We compare our methods with a naive adaptation of BERTQA. We denote the method which continually finetunes the BERTQA model as BASE method.

6.4 Major Results

We compare proposed ProgBERT models with existing continual learning baselines and the results are shown in Table 5 and Table 6. We have the following findings: (1) From Table 5, we can observe that regularization-based approach RegBERTQA achieves significantly better results than BASE methods. The gap in the overall score is about 44, and thus this demonstrates the effectiveness of the regularization-based method on BERTQA. Through compare performance of each domain performance, we observe that RegBERTQA achieves better performance on earlier domains and in the last domain it is worse than the BASE method. This indicates that the regularization-based method indeed hurt the modeling capacity of BERT and prevent the learning from new domains. (2) Second, when comparing four variants of ProgBERT, ProgBERTQA$^{A}$ outperform ProgBERTQA$^{I}$. This validates that inserting adaptor in the BERT layer is more effective than aside fashion. We also observe that ProgBERTQA$^{A-BN}$ outperforms ProgBERTQA$^{A-PAL}$, this demonstrates two layer attention is better than projected attention layer. ProgBERTQA$^{A-BN}$ achieves the best result among these four methods. In next section, we use ProgBERTQA$^{A-BN}$ for analysis and denoted as ProgBERTQA for brevity. (3) When compare ProgBERTQA$^{I-BN}$ with RegBERTQA, we can observe that ProgBERTQA$^{A-BN}$ outperforms RegBERTQA by a large margin. In Table 5, except the first domain, ProgBERTQA$^{A-BN}$ beats ProgBERTQA$^{I-BN}$ on all other domains. This demonstrate the effectiveness of our proposed ProgBERT method.

6.5 Breakdown Analysis

Beyond above overall performance analysis, we also take some breakdown analysis for the CDA task.

---

1 https://github.com/huggingface/transformers

2 https://github.com/mrqa/MRQA-Shared-Task-2019
Table 5: Performance of models on CDA-Q.

| Model                  | wiki | news | scripts | book | tweet | overall |
|------------------------|------|------|---------|------|-------|---------|
| BASE                   | 67.9 | 44.69| 67.8    | 58.51| 85.12 | 324.02  |
| RegBERTQA              | 79.46| 48.98| 62.8    | 60.15| 84.76 | 336.15  |
| ProgBERTQA_{A−PAL}     | 64.59| 49.34| 62      | 56.6 | 80.88 | 313.41  |
| ProgBERTQA_{A−BN}      | 72.13| 56.06| 66.33   | 60.98| 82.94 | 338.44  |
| ProgBERTQA_{I−PAL}     | 67.14| 50.31| 63.09   | 59.46| 83.12 | 323.12  |
| ProgBERTQA_{I−BN}      | 73.66| 56.9 | 67.86   | 64.1 | 85.4  | 347.92  |

Table 6: Performance of models on CDA-C.

6.5.1 Assessing Catastrophic Forgetting. To assess catastrophic forgetting of proposed methods, we show how the performance of the first domain and second domain varies over the training process on the remaining domains [1, 35]. We conduct analysis experiment on CDA-C dataset. We display the results of first domain in Figure 4. Note that we choose the ProgBERTQA_{A−BN} and denote it as ProgBERTQA for brevity. We have following observation: (1) We can see that ProgBERTQA can completely avoid catastrophic forgetting. The reason is that ProgBERTQA fixes the parameters of BERT and domain-specific adapter. However there is no free lunch, ProgBERTQA perform worse than RegBERTQA, when learning the first domain, as the fixed BERT limited the model capacity. (2) From the Figure 4 and 5, we can observe that RegBERTQA still suffer from the catastrophic forgetting. The intuition is that RegBERTQA is with fixed model capacity which theoretically cannot capture all domains. (3) Both ProgBERTQA and RegBERTQA outperform the BASE method, this demonstrate the effectiveness of our proposed two method for CDA for MRC.

6.5.2 Assessing Forward Transfer. In this section, we explore whether ProgBERTQA model can transfer knowledge from the old domain to the new domain. We denotes ProgBERTQA without initialization form previous domain as ProgBERTQA_{init}. We denotes individually fine-tune BERTQA on each domain as INDIVIDUAL. The results are shown in Table 8. (1) We can see that ProgBERTQA outperform INDIVIDUAL in the last four domains. The reason may be INDIVIDUAL learns from each domain and some domain has small size of data. This demonstrate the ProgBERTQA can transfer knowledge from the old domain to the new domain. (2) When we remove adapter initialization from last checkpoint, ProgBERTQA_{init} is worse than ProgBERTQA. This demonstrate the necessity of prior initialization in ProgBERTQA, i.e., initializing new adaptors from last adaptors.

6.5.3 Assessing Order Robustness. In this section, we compare the robustness of each method on the random order of domains. We permute the order of domain and test RegBERTQA and ProgBERTQA.
Table 7: Performance of ProgBRETQA for different adapter size on CDA-Q.

| dim | what | which | where | when | how | why | other | who | overall |
|-----|------|-------|-------|------|-----|-----|-------|-----|---------|
| 32  | 54.20| 55.66 | 65.44 | 74.83| 69.32| 64.07| 57.98 | 80.18| 521.73  |
| 64  | 58.84| 56.73 | 66.31 | 76.21| 69.59| 66.73| 62.10 | 80.45| 537.01  |
| 128 | 58.92| 65.45 | 68.54 | 77.51| 69.68| 64.54| 65.49 | 82.80| 552.96  |
| 256 | 63.14| 70.46 | 67.88 | 79.22| 71.98| 69.46| 65.28 | 83.27| 570.74  |
| 384 | 64.74| 68.98 | 70.74 | 79.31| 73.19| 62.30| 66.11 | 83.40| 568.79  |
| 512 | 64.94| 65.58 | 71.14 | 78.99| 74.83| 63.05| 67.00 | 83.84| 569.41  |

Table 8: Forward transfer analysis.

| method             | wiki | news | scripts | book | tweet | overall |
|--------------------|------|------|---------|------|-------|---------|
| ProgBERTQA         | 75.52| 58.63| 70.51   | 66.62| 86.31 | 357.59  |
| ProgBERTQA_{init}  | 75.52| 56.17| 64.51   | 59.66| 79.81 | 335.67  |
| INDIVIDUAL         | 80.01| 58.14| 66.27   | 58.27| 83.01 | 345.7   |

Table 9: Order robustness analysis of RegBERTQA and ProgBERTQA on CDA-Q.

| order | RegBERTQA | ProgBERTQA |
|-------|-----------|------------|
| 1     | 539.9     | 570.76     |
| 2     | 401.35    | 542.97     |
| 3     | 519.66    | 577.58     |

The results are shown in Table 9. order1 is a random order of domains in CDA-Q, order2 is an ascending order based on the number of training sample in each domain and order3 is a descending order.

From the results we have two observations: (1) Through the performance variance between different orders, we can find that ProgBERTQA is more robust than RegBERTQA. (2) The performance of RegBERTQA degrades severely in order2, as order2 is an ascending order. This is because RegBERTQA is poisoned by domain which has insufficient data or corrupted labels, so RegBERTQA converges to a bad point in parameter space. RegBERTQA cannot escape from that bad converge point, and thus this may cause the order-sensitivity of the RegBERTQA model. (3) ProgBERTQA is more robust compared to RegBERTQA, since it only initializes the adapter with the previous parameters. In conclusion, when there are low-quality data in specific domains, ProgBERTQA is more preferable than RegBERTQA.

6.5.4 Analysis of the Effect of Adapter Size. Adapter size is a hyperparameter in our proposed ProgBERTQA model. Smaller adapter consumes fewer parameters while the performance may decrease. In this section, we test the performance over different adapter size. We tune the adapter size, i.e., the size of the dimension of projection space $d_s$. We vary the adapter size in $\{32, 64, 128, 256, 384, 512\}$ and conduct experiments on both CDA-Q and CDA-C. The results are shown in Table 7 and 10. From the results we see that the 256 is the best adapter size for CDA-Q, and 512 for CDA-C. From Table 7, we can see that when the adapter size exceeds some threshold, the performance gets worse as the adapter gets bigger. A possible reason is the overfitting of adapter. There is a trade-off between the adapter size and the performance. We can control the adapter size to balance the limitation on the number of parameters and the performance requirement.

7 CONCLUSION

We introduce the Continual Domain Adaptation task for MRC. So far as we know, this is the first study on the continual learning perspective of MRC. We build two datasets CDA-Q and CDA-C for the CDA task, by re-organizing existing MRC collections into different domains with respect to the question type and passage type. We conduct preliminary experiments showing the existence of catastrophic forgetting (CF) phenomenon of existing MRC models under the CDA setting. Further, we propose regularization-based RegBERTQA and dynamic-architecture ProgBERTQA to tackle the CDA for MRC. We conduct extensive experiments to analysis the effectiveness of both methods and validate that the proposed dynamic-architecture based model achieves the best performance.

ACKNOWLEDGMENTS

This work was supported by Beijing Academy of Artificial Intelligence (BAAI) under Grants No. BAAI2019ZD0306 and BAAI2020ZJ0303, and funded by the National Natural Science Foundation of China (NSFC) under Grants No. 61722211, 61773362, 61872338, and 61902381, the Youth Innovation Promotion Association CAS under Grants No. 20144310, and 2016102, the National Key RD Program of China under Grants No. 2016QY02D0495, the Lenovo-CAS Joint Lab Youth Scientist Project, the K.C.Wong Education Foundation, and the Foundation and Frontier Research Key Program of Chongqing Science and Technology Commission (No. cstc2017jcyjBX0059).
REFERENCES

[1] Taneem Adel, Han Zhao, and Richard E. Turner. 2020. Continual Learning with Adaptive Weights (CLAW). ArXiv abs/1911.09514 (2020).

[2] Shibha Anghar, Lilly Mou, Kim A Selby, Kevin D Fantasdo, Pascal Poupart, and Xin Jiang. 2018. Progressive memory banks for incremental domain adaptation. arXiv preprint arXiv:1811.00259 (2018).

[3] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2017. Layer Normalization. arXiv abs/1607.06450 (2016).

[4] Arslan Chaudhry, Puneet Kumar Dokania, Thalaiyasingam Ajanthan, and Philip S. Torr. 2018. Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence. In ECCV.

[5] Danqi Chen. 2018. Neural Reading Comprehension and Beyond. Ph.D. Dissertation. Stanford University.

[6] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In Association for Computational Linguistics (ACL).

[7] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to Answer Open-Domain Questions. In ACL.

[8] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAc: Question Answering in Context. ArXiv abs/1808.07036 (2018).

[9] Jaime A. Collazo, Adam J Terando, Augustin C. Engman, Paul F. Fackler, and Thomas Kwak. 2018. Toward a Resilience-Based Conservation Strategy for Wetlands in Pueto Rico: Meeting Challenges Posed by Environmental Change. Wetlands (2018), 1–15.

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

[11] Prithviraj Dhar, Rajat Vrikram Singh, Kuan-Chuan Peng, Ziyun Wu, and Rama Chellappa. 2019. Learning Without Memorizing. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019). p. 5133–5141.

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

[13] Dan Hendrycks and Kevin Gimpel. 2016. Gaussian Error Linear Units (GELUs). arXiv Learning (2016).

[14] Karl Moritz Hermann, Tomáš Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In NIPS.

[15] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Antonio González, Maximilian Schmitt, and Sylvain Gelly. 2016. Parameter-Efficient Transfer Learning for NLP. In ICML.

[16] Hsin-Yuan Huang, Chengchung Zhu, Yelong Shen, and Weizhu Chen. 2018. FusionNet: Fusing via Fully-Aware Attention with Application to Machine Comprehension. ArXiv abs/1711.07341 (2018).

[17] Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel D. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving Pre-training by Representing and Predicting Spans. Transactions of the Association for Computational Linguistics 8 (2020), 64–77.

[18] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimisation. CoRR abs/1412.6980 (2015).

[19] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Guillaume Desjardins, Andrew A Rusu, Ruslan Salakhutdinov, Oriol Vinyals, Tomasz Zawadzki, Daan Wierstra, and Yoshua Bengio. 2017. Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences 114, 13 (2017), 3521–3526.

[20] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Guillaume Desjardins, Andrew A Rusu, Ruslan Salakhutdinov, Oriol Vinyals, Tomasz Zawadzki, Daan Wierstra, and Yoshua Bengio. 2017. Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences of the United States of America 114, 13 (2017), 3521–3526.

[21] Tomáš Kociský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA Reading Comprehension Challenge. Transactions of the Association for Computational Linguistics 6 (2018), 317–328.

[22] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Ilia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: A Benchmark for Question Answering Research. Transactions of the Association for Computational Linguistics 7 (2019), 453–466.

[23] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. 2017.Race: Large-scale Reading Comprehension Dataset From Examinations. In EMNLP.