Plug-and-Play Adaptation for Continuously-updated QA

Kyungjae Lee[5] **Wookje Han**[1] **Seung-won Hwang**[1]*
Hwaran Lee[2] **Joonsuk Park**[2,4] **Sang-Woo Lee**[2,3]
[1]Seoul National University [2]NAVER AI Lab [3]NAVER CLOVA
[4]University of Richmond [5]LG AI Research

Abstract

Language models (LMs) have shown great potential as implicit knowledge bases (KBs). And for their practical use, knowledge in LMs need to be updated periodically. However, existing tasks to assess LMs' efficacy as KBs do not adequately consider multiple large-scale updates. To this end, we first propose a novel task—Continuously-updated QA (CuQA)—in which multiple large-scale updates are made to LMs, and the performance is measured with respect to the success in adding and updating knowledge while retaining existing knowledge. We then present LMs with plug-in modules that effectively handle the updates. Experiments conducted on zsRE QA and NQ datasets show that our method outperforms existing approaches. We find that our method is 4x more effective in terms of updates/forgets ratio, compared to a fine-tuning baseline.

1 Introduction

LM-as-KB is a new paradigm in which pre-trained language models (LMs) are used as implicit knowledge bases (KBs) (Petroni et al., 2019). This is made possible by LMs' impressive ability to memorize factual knowledge (Heinzerling and Inui, 2021; Brown et al., 2020). Recently, two tasks have been used to assess such ability: LAMA, a knowledge probing benchmark, challenges LMs to fill in masked words over relational knowledge (Petroni et al., 2019); and closed-book QA (CBQA) examines whether LMs can correctly answer natural language questions (Roberts et al., 2020).

For practical usage, LM-as-KB requires that LMs are updated periodically to stay current with the ever-evolving world. Thus, LMs' ability to update knowledge should also be evaluated. To this end, we present **Continuously-updated QA (CuQA)**, which tests the ability to continuously inject knowledge to update (or **target knowledge**),

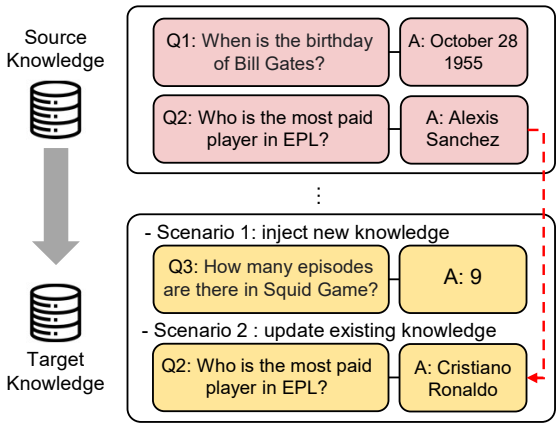

Figure 1: Examples of CuQA showcasing two scenarios.

while retaining existing knowledge (or **source knowledge**). Specifically, we consider multiple large-scale knowledge updates (8k to 60k) covering two scenarios: injecting new knowledge (Scenario 1 in Figure 1) and updating existing knowledge (Scenario 2 in Figure 1) .

Our goal is to organize the implicit storage of knowledge, to add target knowledge (*yellow box* in Figure 1) and anchor to select target knowledge. A simple approach is to train updated LMs from scratch; however, this is far too expensive considering the parameter sizes of recent LMs, such as 175B for GPT-3 (Brown et al., 2020) and about 11B for T5 (Raffel et al., 2020). There has also been related work for the two scenarios. For Scenario 1, a method for continual learning can be adopted, constraining the distance between parameters before and after fine-tuning (Chen et al., 2020). However, this approach still suffers from so-called catastrophic forgetting, where the LMs fail to retain large amounts of source knowledge. For Scenario 2, one may consider knowledge editing methods, where we see reasonable performances for a single knowledge edit while retaining the rest (De Cao et al., 2021; Mitchell et al., 2021). However, this line of work does not perform well when multiple

*correspond to seungwonh@snu.ac.kr

Findings of the Association for Computational Linguistics: ACL 2022, pages 438 - 447

edits are accumulated, *e.g.*, only 67% of 125 edits were updated, as reported in (Mitchell et al., 2021).

We propose to efficiently extend LMs with plug-and-play modules that store target knowledge. More specifically, we adopt a parameter-expansion method in which the LM storing existing knowledge is extended with plug-in feed-forward modules storing updated knowledge. Depending on the input, the LM selectively uses either the original LM or a plug-in module. We stress that, by keeping the original LM intact, we retain (a) not only source knowledge, (b) but also those outdated from updates (*red arrow* in Figure 1). (a) is important to avoid catastrophic forgetting, while (b) is useful when updates need to be reverted due to ethical concerns—for example, there can be malicious attempts to override facts.

We evaluate our approach on zsRE (Levy et al., 2017) and Natural Questions (Kwiatkowski et al., 2019) to showcase successful updates of new knowledge and retention of existing knowledge. We measure the accuracies on both previous and updated knowledge and find that ours show x4 higher updates/forgets ratio, compared to fine-tuning. We also release our code and dataset.[1]

Our key contributions are as follows:

- We present CuQA, a novel task to assess LMs' ability to continuously inject knowledge to update.
- We propose a new methodology, plug-and-play adaptation, to continually learn new knowledge while better retaining existing knowledge.

2 Related Work

The relevant research can be categorized into three groups: Knowledge Editing, Continual Learning, and Adaptation. In Table 1, we compare these with our method.

Editing Implicit Knowledge In Table 1(a), knowledge editing methods (De Cao et al., 2021; Mitchell et al., 2021; Dai et al., 2021) aim to efficiently edit model's parameters on examples that have conflicts with old facts, while preserving the outputs of untargeted examples. Instead of directly updating gradients by fine-tuning, these methods transform the gradients for new edit parameters. As representative methods for knowledge editing, KnowledgeEditor (KE) (De Cao et al., 2021) using LSTM produces gate vectors, then the gated sum of gradients is updated into the model, while MEND (Mitchell et al., 2021) uses simple MLP layers and residual connections for the same purpose. Although these methods succeeded in updating the target examples less forgetting, their target scenario is a single edit, such that the cumulative effect of multiple edits does not reflect well, which disqualifies its use for our target task of update large-scale data (8K∼60K). As reported in (Mitchell et al., 2021), MEND successfully updates only 67% of edits when applying 125 edits, while our finding was consistent when none of the 125 edits was applied in our evaluation.[2] In addition, for editing previous knowledge, KE and MEND simulate knowledge updates, by generating synthetic knowledge from LM. Such generations may not be realistic data and also give unfair advantages to LM-based methods, while we use actual up-to-date knowledge as new data, which were annotated on recent corpus (Zhang and Choi, 2021).

Method	Forgetting less?	Scales to a large set?	Conflict with old facts?
(a) Editing	✓	✗	✓
(b) CL	✓	✓	✗
(c) Adaptation	✗	✓	✗
Our Method	✓	✓	✓

Table 1: Conceptual comparison of existing approaches.

Continual Learning (CL) for NLP For our task, we can adopt CL methods, learning a new task while preserving the accuracy on previous tasks. Kirkpatrick et al. (2017) proposed Elastic Weight Consolidation, alleviating catastrophic forgetting. This method regularizes learning on a new task, by constraining the parameters trained on the previous task. For NLP tasks, RecAdam (Chen et al., 2020) uses the regularization and annealing technique, which is a CL baseline in our experiment. While CL approaches focusing on forgetting do not consider conflicts between old and new knowledge, our work deals with such a realistic scenario. Additionally, previous work (Dhingra et al., 2021) proposed benchmarks for probing temporal language models, asking "Fill-in-the-Blank (FIB)" questions. Meanwhile, FIB questions are limited to evaluate masked language models, such as BERT and RoBERTa. We extend to evaluate arbitrary questions for a knowledge-intensive task; closed-

[1] https://github.com/wookjeHan/Continual-Plug-and-Adapt-for-CuQA/

[2] In the case of KE, we reimplement the released code for testing: https://github.com/nicola-decao/KnowledgeEditor.

book QA, which can evaluate generative LMs with broader applicability, to include T5 and GPT.

Task-aware Adaptation for Transformers Recent works (Hu et al., 2021; Wang et al., 2020; Lin et al., 2020) study LM adaptation to new labeled data in a new domain, which has a different data distribution from that at pretraining. These works show performance improvements on downstream tasks in the new domain, while fine-tuning a small number of parameters. However, these adaptation methods do not consider sequential training, and overwrite the new data into the parameters that store previous knowledge. In our experiment, it is observed that the adaptation methods are rapidly forgetting previously seen data, while performing well on new knowledge.

3 A Continuously-updated QA Task

Task Description In this section, we propose Continuously-updated QA (CuQA), a new continual learning task for knowledge updates in LMs based on closed-book QA (CBQA) (Roberts et al., 2020). In CBQA, LMs answer factual questions with the implicit knowledge stored in the model, without any external context (*i.e.*, in contrast to open-domain QA), so that LMs are required to adequately update their parameters to the target knowledge. In our CuQA, LMs learn source (original) knowledge first, then update them with target (new) knowledge without source knowledge access. For the above setting, source knowledge (to be retained) and target knowledge (to be added) in CuQA do not have any overlap of QA pairs (or paraphrases) for any given fact.

Specifically, we denote a factual pair of question and answer as (q, a), source knowledge as $\mathcal{K}_s$, and target as $\mathcal{K}_t$. We first build an initial model θ^{old} pre-trained on source knowledge $\mathcal{K}_s$. Then, we inject target knowledge $\mathcal{K}_t$ into the pre-trained model and obtain the infused model θ^{new}. Our goal is to memorize $\mathcal{K}_t$ on model θ^{new}, with less forgetting $\mathcal{K}_s$. If knowledge in $\mathcal{K}_t$ conflicts one in $\mathcal{K}_s$, the model is required to adjust its parameters by reflecting the target knowledge. Note that multiple target knowledge can be sequentially updated to the model (see details in Section 4).

Research Questions CuQA is designed to address the following research questions:

- RQ1: Can the method learn target knowledge while retaining source knowledge?
- RQ2: How does sequentially learning multiple target knowledge affect the performance?
- RQ3: How does the size of each target knowledge affect the performance?

Metric For evaluation, we measure the success of updates, retaining of source knowledge, and generality using exact match (EM) scores. Additionally, we measure the ratio of forgets to updates.

- **Accuracy on** $\mathcal{K}_t$: we evaluate how much model θ^{new} **successfully updates** examples in $\mathcal{K}_t$.
- **Accuracy on** $\mathcal{K}_s$: how much model θ^{new} **forgets** examples in $\mathcal{K}_s$. This indicates performance degradation, when replacing θ^{old} with θ^{new}.
- **Accuracy on** $\mathcal{P}_s$**,** $\mathcal{P}_t$: how well model θ^{new} **generalizes** on semantically equivalent questions (or paraphrases).
- **F/U Ratio** (# of forgets/# of updates): how many examples in $\mathcal{K}_s$ are forgotten per an update of one example in $\mathcal{K}_t$. (# of forgets) is equal to the difference of correct prediction cases in $\mathcal{K}_s$, between θ^{old} and θ^{new}.

4 Method

In this section, we describe baseline approaches (Section 4.1), and introduce our proposed method for plug-and-play adaptation (Section 4.2).

4.1 Baseline Approaches

We establish three baseline for (a), (b), and (c), in Table 1. Since we found that a knowledge editing approach is outperformed by fine-tuning, we exclude it as baselines, and add fine-tuning instead.

Fine-tuning on target knowledge As a naive baseline, we start with the previous work (Roberts et al., 2020) for CBQA, by fine-tuning T5 (Raffel et al., 2020) with encoder-decoder structure. This baseline is to fine-tune the pre-trained model θ^{old} on facts in $\mathcal{K}_t$ to minimize the loss:

$$\mathcal{L}_{FT} = \sum_{(q,a)\in\mathcal{K}_t} L((q,a);\theta) \quad (1)$$

where L refers to a seq2seq loss. This baseline is expected to optimize accuracy on target knowledge $\mathcal{K}_t$, thus increases the distance between the before- (θ^{old}) and after-parameters (θ^{new}) resulting in the risk of forgetting. For other baselines and our method, we adopt the same transformer: T5 as backbone network.

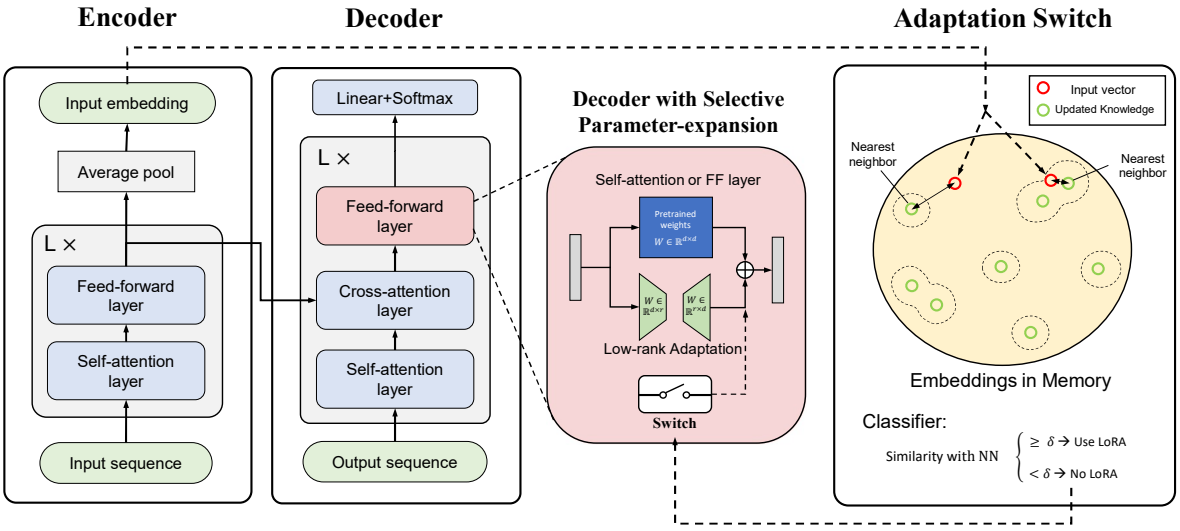

Figure 2: An overview of our proposed architecture.

Regularized fine-tuning for CL We adopt RecAdam (Chen et al., 2020) aiming to reduce the forgetting risk by adding a constraint to minimize the distance between θ^{old} and θ^{new} as follow:

$$\mathcal{R} = \|(\theta - \theta^{old})\|_p \quad (2)$$

where $\|\cdot\|_p$ indicates L_p norm. In addition, RecAdam uses an annealing technique, controlling the ratio between $\mathcal{R}$ and the fine-tuning loss (Eq. (1)) as follows:

$$\mathcal{L}_{total} = \lambda(t)\mathcal{L}_{FT} + (1 - \lambda(t))\mathcal{R}, \quad (3)$$

$$\lambda(t) = \frac{1}{1 + \exp(-k \cdot (t - t_0))} \quad (4)$$

where k and t_0 are hyper-parameters.

Adapters for knowledge updates For adaptation approaches, we implement two parameter-expansion methods: K-adapter (Wang et al., 2020) and LoRA (Hu et al., 2021). The approaches freeze the parameters θ^{old} in pre-trained LM and augment additional new parameters $\tilde{\theta}$ in the LM to train target knowledge as following:

$$\mathcal{L}_{adap} = \sum_{(q,a)\in\mathcal{K}_t} L((q,a); \theta^{old}, \tilde{\theta}). \quad (5)$$

For $\tilde{\theta}$, K-adapter (Wang et al., 2020) uses augmented self-attention layers, while LoRA (Hu et al., 2021) utilizes extra low-rank matrices.

4.2 Our Method

Motivated by the intuition of regularization to preserve source knowledge and that of adapters to inject target knowledge into new parameters, we show their strengths can be combined for our task. At the inference phase, our method selectively uses the plug-in modules to keep source knowledge intact, while tasks requiring target knowledge will be redirected to new plug-in modules.

Specifically, our distinction is augmenting function f (in an original LM) with function g, representing source and target knowledge respectively. The function f is a single layer in transformer trained on source knowledge $\mathcal{K}_s$, and g is an augmented function with new parameters for $\mathcal{K}_t$. Existing work, such as LoRA, can be interpreted by adding the two functions:

$$h = f(x) + g(x) \quad (6)$$

where f is one-linear layer in self-attention or feed-forward layers. That is, $f(x) = W_0 x$, where $W_0 \in \mathrm{R}^{d\times k}$ denotes the pre-trained and fixed parameters. LoRA uses low-rank matrices as $g(x)$, *i.e.*, $g(x) = BAx$, where $B \in \mathrm{R}^{d\times r}$, $A \in \mathrm{R}^{r\times k}$, and $r \ll min(d, k)$. The low-rank matrices A and B are trainable parameters for updating target knowledge. The new layer with the additional matrices is denoted as follows:

$$h = W_0 x + BAx = (W_0 + BA)x \quad (7)$$

However, the above add-aggregation has a limitation, as $g(x)$ can affect the model's outputs, and increase the distance between hidden states in θ^{old} and θ^{new}, which causes a forgetting problem.

Our key distinction is adding a selector, that is selectively activated for q requiring the use of plug-

in module g, as follows:

$$h = f(x) + \sigma(q) \cdot g(x) \tag{8}$$

where $\sigma(q)$ is 1 or 0 depending on query q. While there can be various ways to train the selector in a sophisticated way, supervised either directly, or indirectly in an end-to-end manner, we show a simple unsupervised selector is already sufficient to show gains. Specifically, our selector is a key-value lookup where the key is m_i and value is g. At inference time, when given query q is based on facts in $\mathcal{K}_t$, we activate the augmented g for generating its output. If q is not from $\mathcal{K}_t$, we use only the original model θ^{old} for generation. To classify whether the input is from $\mathcal{K}_t$ or not, we build explicit memory with embeddings of $\mathcal{K}_t$ and leverage the distance with nearest neighbor (NN) in the memory.

Let $\mathcal{M} \in \mathrm{R}^{N \times d}$ be memory embeddings that stores embeddings of input questions in $\mathcal{K}_t$, where N is the total number of examples in $\mathcal{K}_t$. As shown in Figure 2, question embedding can be extracted from the encoder, by averaging the hidden states of input sequence. In T5 model with encoder-decoder, this averaging method is known to be effective on semantic textual similarity, as in (Ni et al., 2021). Given question q, cosine similarity with NN is calculated as follows:

$$s_q = \max_i(\mathrm{sim}(m_i, q)), \quad m_i \in \mathcal{M} \tag{9}$$

where sim indicates cosine similarity. Based on s_q, if the score is greater than or equal to threshold δ, we assume q is from target knowledge $\mathcal{K}_t$. We build a indicator function as follows:

$$\sigma(q) = \begin{cases} 1 & \text{if } s_q \geq \delta, \\ 0 & \text{if } s_q < \delta. \end{cases} \tag{10}$$

In other words, $s_q \geq \delta$ indicates that input q is semantically similar with one fact in $\mathcal{K}_t$. At that time, our model is augmented with g that stores new and updated knowledge.

Meanwhile, as shown in Figure 2, we apply the selective use of parameters to only a decoder in a transformer architecture, not a encoder. The switch σ depends on query embedding q, and the embedding q is extracted from T5 encoder. If we apply the switch σ to hidden states in T5 encoder, this causes a recursion relation, or inefficient computations. By augmenting g for the decoder, embedding q is not changing during updating target knowledge, and depends on only pre-trained θ^{old}.

General case of multiple knowledge updates Our new perspective has another benefit of naturally generalizing to sequential (>2) sources. Assume that there are multiple target knowledge to be sequentially updated, *i.e.*, $\mathcal{K}_t^1, \mathcal{K}_t^2, ..., \mathcal{K}_t^M$. We build multiple functions g_k and memories $\mathcal{M}_k$ (where $k = 1, ...M$), according to each target knowledge. The new function considering the multiple knowledge is denoted as follows:

$$h = f(x) + \sum_{k=1}^{M} \sigma_k(q) \cdot g_k(x) \tag{11}$$

During training j-th target $\mathcal{K}_t^j$, the switch $\sigma_k(q)$ is activated where $1 \leq k \leq j$. At inference time, our selector extracts top1-NN fact m^*, which is closest to a query q. If m^* is in $\mathcal{M}_k$, the switch $\sigma_j(q)$ is activated where $1 \leq j \leq k$, as follows:

$$m^* = \underset{m}{\mathrm{argmax}}(\mathrm{sim}(m, q)), \quad m \in \mathcal{M}_{1:M} \tag{12}$$

If the NN fact m^* is in $\mathcal{M}_j$, we estimate that its implicit knowledge is stored in the accumulated function $\sum_{k=1}^{j} g_k(x)$. That is, when m^* is in $\mathcal{M}_j$, the activation is decided as follows:

$$\sigma_k(q) = \begin{cases} 1 & \text{if } s_q \geq \delta \text{ and } 1 \leq k \leq j, \\ 0 & \text{if } s_q < \delta. \end{cases} \tag{13}$$

An alternative adapter We can replace LoRA with K-adapter (Wang et al., 2020). In K-adapter, f is a transformer layer (denoted as $\mathrm{TRM}(x)$), and g is multiple transformer layers with two projection layers (denoted as $\mathrm{KIA}(x)$)). That is, $f(x) = \mathrm{TRM}(x)$, consisting of one self-attention & two feed-forward layers. In the original paper (Wang et al., 2020), $g(x)$ consists of multiple transformer layers and up&down projection layers. For K-adapter, we set a simple version with only a single transformer layer, as follows:

$$h = \mathrm{TRM}(x) + \mathrm{KIA}(x) \tag{14}$$

where the parameters in TRM are fixed and that in KIA is trainable on target knowledge.

5 Experiment

In this section, we demonstrate the effectiveness of our approach on CuQA.

Datasets We evaluate our method on the following closed-book QA datasets:

(1) **Zero-shot Relation Extraction (zsRE)**: Levy et al. (2017) build relation-specific QA pairs, and De Cao et al. (2021) utilize this dataset for a closed-book QA task. This set provides question paraphrases based on the same fact and answer. We split this set into two groups ($\mathcal{K}_s$ and $\mathcal{K}_t$) that do not share the same facts. To validate generalization, we build held-out sets ($\mathcal{P}_s$ and $\mathcal{P}_t$) that are not used in training process. For this, we sample one QA pair among paraphrases based the same fact as $\mathcal{P}$.

(2) **Natural Questions (NQ) + SituatedQA**: Kwiatkowski et al. (2019) build NQ – a large-scale QA dataset based on user queries. We consider NQ as source knowledge $\mathcal{K}_s$ except outdated facts based on SituatedQA. Zhang and Choi (2021) proposed SituatedQA identifying temporal- and geographical-dependent questions on a subset of NQ. We use the temporal-dependent QA pairs as $\mathcal{K}_t$, which are annotated based on 2021 dump of Wikipedia. For $\mathcal{P}_s$ and $\mathcal{P}_t$, as both NQ and SituatedQA do not provide paraphrases, we follow (De Cao et al., 2021) using back-translation for generating paraphrases.

Implementation For T5 model, we use a large version with total 770M parameters. In our experiment, we assume that the old model θ^{old} storing source knowledge is available. For NQ, we used the open-source pre-trained model[3] as the model θ^{old}. For zsRE, we load and train T5 model[4] on source knowledge. For training, we set batch size 64 on 4 RTX3090 GPUs, and used Adam (Kingma and Ba, 2015) optimizer with learning rate 4e-4. For development set, we sample each 1K from $\mathcal{K}_s$, $\mathcal{K}_t$, and select the maximum harmonic mean of their accuracies as a best model. As a hyper-parameter, we search δ in a range of [0,1] with 0.05 step size, and found the best value (δ=0.9) based on development set. As embedding memory $\mathcal{M}$, we used additional parameters: 60M for zsRE and 8.5M for NQ. The size of the memories can be reduced by several techniques, such as random projection (Luan et al., 2020) and binary encoding (Yamada et al., 2021), which is left out of our focus.

Comparison with baselines We compare our method with baselines, as mentioned in Section 3.2; Fine-tuning (B-I), RecAdam (B-II), LoRA (B-III), and K-adapter (B-IV). When re-implementing K-adapter, we do not freeze the parameters of decoder, unlike in the original paper (Wang et al., 2020), because the performance is not changing when freezing. We train each model until 80 epochs and select a best model by the harmonic mean of source/target knowledge in development set.

[3]https://huggingface.co/google/t5-large-ssm-nq
[4]https://huggingface.co/google/t5-large-ssm

	The total # of examples			
	$\mathcal{K}_s$	$\mathcal{P}_s$	$\mathcal{K}_t$	$\mathcal{P}_t$
zsRE (Large)	60K	24K	60K	24K
zsRE (Medium)	60K	24K	30K	12K
zsRE (Small)	60K	24K	15K	6K
NQ + SituatedQA	59K	32K	8.3K	1.6K

Table 2: Statistics of datasets.

5.1 R1: Comparing Ours with Baselines

Table 3 shows our main experimental results on two CBQA datasets. First, the model θ^{old} memorizes the source knowledge $\mathcal{K}_s$ well and generalizes on the paraphrase set $\mathcal{P}_s$ as well, showing high accuracy on both datasets. After training on $\mathcal{K}_t$, all models perform well on $\mathcal{K}_t$ and $\mathcal{P}_t$. These results indicate that these models are at least appropriate for memorizing training data in the current task.

Meanwhile, while acquiring $\mathcal{K}_t$, the models show variant results on $\mathcal{K}_s$ and $\mathcal{P}_s$, which have the different ability of retaining previous knowledge against forgetting. In Fine-tuning (B-I), its performances on source knowledge $\mathcal{K}_s$ and $\mathcal{P}_s$ decrease as training epochs (see Figure 3). RecAdam (B-II) alleviates the forgetting problem of fine-tuning, but the performance gains are marginal on two datasets. K-adapter (B-III) shows the strong performance on $\mathcal{K}_s$ with less forgetting, however, does not perform well on $\mathcal{P}_s$ and $\mathcal{P}_t$ showing low generalization. Because LoRA (B-IV) has the fewest trainable parameters, its forgetting is more aggravated, showing the worst performance on $\mathcal{K}_s$ and $\mathcal{P}_s$ in both zsRE and NQ. Ours with either K-adapter or LoRA shows the best performance on $\mathcal{K}_s$ and $\mathcal{K}_t$. In terms of the F/U ratio, our method also shows the lowest loss when updating one new example. Figure 3 shows how the performance of each model changes over training epochs, on the development set.

Ablation study In an ablation study, we test which component has the higher impact on memorizing implicit knowledge, on paraphrase set $\mathcal{P}_s$ and $\mathcal{P}_t$. In our method with LoRA, the function f in Eq. (8) can be applied to any pro-

Method	# of Prams (train/total)	zsRE Question Answering					NQ (with SituatedQA)				
		$\mathcal{K}_s$	$\mathcal{P}_s$	$\mathcal{K}_t$	$\mathcal{P}_t$	F/U Ratio	$\mathcal{K}_s$	$\mathcal{P}_s$	$\mathcal{K}_t$	$\mathcal{P}_t$	F/U Ratio
Model θ^{old}	-	95.6	95.2	25.7	28.5	-	96.6	94.9	35.3	33.7	-
B-I: Fine-tuning	737M / 737M	76.7	70.6	92.6	85.9	0.284	92.9	82.5	94.9	**92.9**	0.435
B-II: RecAdam	737M / 737M	80.5	74.7	91.6	83.5	0.230	93.1	82.1	93.8	92.1	0.419
B-III: K-adapter	538M / 840M	80.5	70.8	**96.4**	89.6	0.215	94.4	81.4	94.8	89.4	0.259
B-IV: LoRA	62M / 799M	71.1	62.9	92.9	84.8	0.366	89.8	74.0	94.0	90.5	0.800
Ours (+K-adapter)	538M / 840M	86.3	78.9	**96.4**	**91.1**	0.132	**95.6**	88.1	94.9	90.3	0.118
Ours (+LoRA)	62M / 799M	**90.5**	**90.6**	95.3	89.4	**0.073**	**95.6**	**95.2**	**95.1**	90.0	**0.117**

Table 3: The comparison of the continual learning results on zsRE (Large) and NQ datasets. We measure the accuracies on the knowledge $\mathcal{K}_s, \mathcal{K}_t$, and the paraphrase knowledge $\mathcal{P}_s, \mathcal{P}_t$, with the F/U ratio.

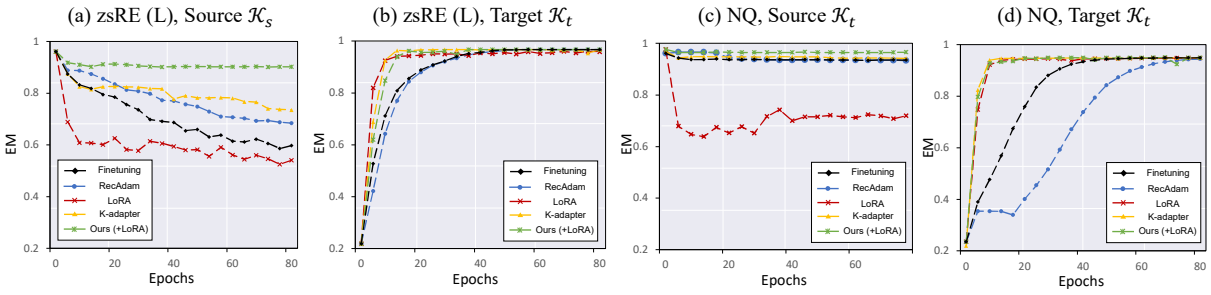

Figure 3: Accuracies of ours and baselines over training epochs.

Type	W_Q,W_V			W_{FF}			All
Rank r	16	64	256	16	64	256	256
$\mathcal{P}_s$	94.9	95.2	95.5	95.1	95.0	95.2	95.2
$\mathcal{P}_t$	59.6	65.1	65.5	87.1	89.2	90.0	89.3

Table 4: An ablation study

jection layer in transformers. While the original work (Hu et al., 2021) applies to query- and value-matrices (W_Q, W_V) in self-attention, we consider feed-forward layers (W_{FF}), as well as self-attention. In addition, we observe how does the performance vary when the number of parameters increases by controlling rank r. In Table 4, we empirically found applying feed-forward layers is more effective than query and value projection, especially on target knowledge $\mathcal{P}_t$. These results indicate that memorizing factual knowledge is more relevant with a feed-forward module, which is consistent with the views in (Sukhbaatar et al., 2019; Geva et al., 2020).

5.2 R2: Accumulating over Multiple $\mathcal{K}_t$

To evaluate the scalability of our method on multiple $\mathcal{K}_t$ (>2), we assume multiple updates (five-phase) with smaller amount of examples, by splitting target knowledge $\mathcal{K}_t$ in zsRE (Large, 60K), into four sets, from $\mathcal{K}_t^1$ to $\mathcal{K}_t^4$ (each 15K). In this experiment, we train models during 40 epochs/phase. To generalize for LoRA baseline, we aggregate multiple g_k by addition, by activating all the switches at inference, *i.e.*, $\sigma_{1:M}(x) = 1$ in Eq. (13). This setting assumes that this baseline cannot leverage our selector to organize the storage of implicit knowledge. Figure 4 shows the performances of Fine-tuning, LoRA, and Ours, over training epochs. In fine-tuning, the accuracy on source knowledge keeps dropping during the whole training process. In LoRA, multiple updating deteriorates memorizing target knowledge stored in adapters, faster than source knowledge stored in the original parameters. This indicates that the fewer parameters, the faster the forgetting. In contrast, our method consistently outperforms the baselines, by retaining five knowledge, with forgetting less. To summarize these results, sequential updates aggravate forgetting of the fine-tuning method, which can be overcome through the selective use of adapters.

5.3 R3: Over varying Size of $\mathcal{K}_t$

As the size of target knowledge increases, it makes LMs suffer from more forgetting, increasing the

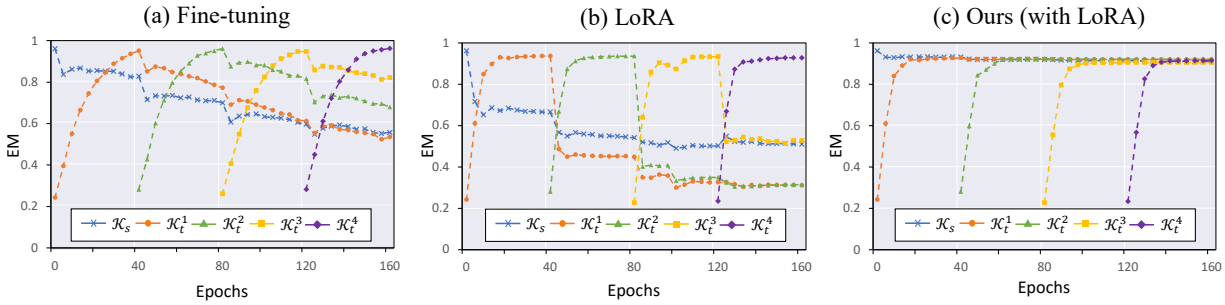

Figure 4: The accuracies on multiple knowledge sources (K=5) over training epochs for zsRE.

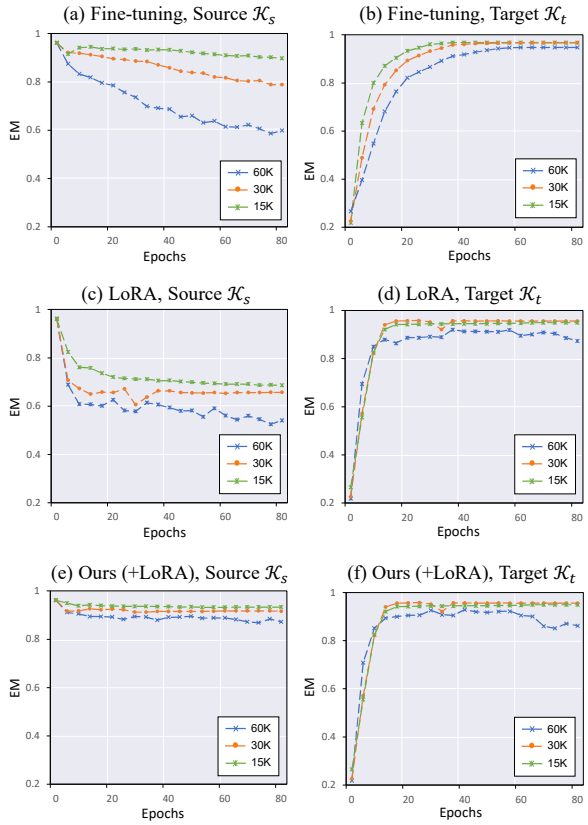

Figure 5: Accuracies over varying size of zsRE.

distance between before- and after-parameters. In this section, we observe how does the performance of each model vary as different sizes of $\mathcal{K}_t$. Figure 5 shows the the accuracies of zsRE datasets (Large-60K, Medium-30K, Small-15K), over training epochs. On source knowledge $\mathcal{K}_s$, the performance of fine-tuning and LoRA keeps dropping, and the accuracy drops are proportional to the size of target knowledge. Meanwhile, our method with LoRA consistently maintains high performance, which is not sensitive to training epochs. On target knowledge $\mathcal{K}_t$, the performances of three models reach high accuracy. However, our method on Large zsRE shows unstable performance at the end

		Ground-truth	
		Source	Target
Selector Prediction	Source	19527 (40.7%)	854 (1.8%)
	Target	4473 (9.3%)	23146 (48.2%)

Table 5: The confusion matrix of Selector.

		Ground-truth	
		Source	Target
Selector Prediction	Source	95.3	35.1
	Target	70.8 (0.0)	91.7 (97.4)

Table 6: The accuracies of Ours/Retrieval in four cases.

of training, which may need to use early stopping.

5.4 Analysis of Selector

In Table 5, we show the distribution of selector's predictions and the ground-truths, in our experiment on zsRE (Large). Nearest Neighbor-based selector successfully classifies 88.9% of examples, while 11.1% failed. In our method, if the selector classifies an input as target knowledge, the plug-in g is activated. Instead of the use of g, we can retrieve answers aligned with questions in $\mathcal{M}$, not generate them. We compare our generation with the retrieval in each case of Table 5. Table 6 shows the accuracy of predicting the answers, where the numbers in each cell indicate EM of our generation (retrieval: in parentheses). If an example in source knowledge is incorrectly classified as target, there is no relevant fact in $\mathcal{M}$, thus the accuracy in this case is zero. In contrast to Retrieval, our generative method is robust in this case, achieving 70.8% EM, because ours with g learned the source knowledge.

6 Conclusion

This paper studies how to accumulate new knowledge to LMs that stores existing knowledge. We propose a simple yet effective method to update target knowledge into new parameters, preventing from forgetting source knowledge. On two datasets: zsRE and NQ, our empirical results show that our proposed method can improve existing approaches for continual learning or task adaptation.

7 Acknowledgement

This research was supported by SNU-NAVER Hyperscale AI Center, and IITP grants funded by the Korea government (MSIT) [2021-0-02068 SNU AIHub, IITP-2022-2020-0-01789].

References

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

Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7870–7881.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, and Furu Wei. 2021. Knowledge neurons in pretrained transformers. *arXiv preprint arXiv:2104.08696*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Bhuwan Dhingra, Jeremy R Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W Cohen. 2021. Time-aware language models as temporal knowledge bases. *arXiv preprint arXiv:2106.15110*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2020. Transformer feed-forward layers are key-value memories. *arXiv preprint arXiv:2012.14913*.

Benjamin Heinzerling and Kentaro Inui. 2021. Language models as knowledge bases: On entity representations, storage capacity, and paraphrased queries. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1772–1791.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114(13):3521–3526.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342.

Zhaojiang Lin, Andrea Madotto, and Pascale Fung. 2020. Exploring versatile generative language model via parameter-efficient transfer learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 441–459.

Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. 2020. Sparse, dense, and attentional representations for text retrieval. *arXiv preprint arXiv:2005.00181*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. 2021. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426.

Sainbayar Sukhbaatar, Edouard Grave, Guillaume Lample, Herve Jegou, and Armand Joulin. 2019. Augmenting self-attention with persistent memory. *arXiv preprint arXiv:1907.01470*.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Guihong Cao, Daxin Jiang, Ming Zhou, et al. 2020. K-adapter: Infusing knowledge into pre-trained models with adapters. *arXiv preprint arXiv:2002.01808*.

Ikuya Yamada, Akari Asai, and Hannaneh Hajishirzi. 2021. Efficient passage retrieval with hashing for open-domain question answering. *arXiv preprint arXiv:2106.00882*.

Michael JQ Zhang and Eunsol Choi. 2021. Situatedqa: Incorporating extra-linguistic contexts into qa. *arXiv preprint arXiv:2109.06157*.