CorpusBrain++: A Continual Generative Pre-Training Framework for Knowledge-Intensive Language Tasks

JIAFENG GUO, CHANGJIANG ZHOU, RUQING ZHANG, and JIANGUI CHEN, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China and University of Chinese Academy of Sciences, Beijing, China
MAARTEN DE RIJKE, University of Amsterdam, Amsterdam, The Netherlands
YIXING FAN and XUEQI CHENG, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China and University of Chinese Academy of Sciences, Beijing, China

Knowledge-intensive language tasks (KILTs) typically require retrieving relevant documents from trustworthy corpora, e.g., Wikipedia, to produce specific answers. Very recently, a pre-trained generative retrieval model for KILTs, named CorpusBrain, was proposed and reached new state-of-the-art retrieval performance. However, most research on KILTs, including CorpusBrain, has predominantly focused on a static document collection, overlooking the dynamic nature of real-world scenarios, where new documents are continuously being incorporated into the source corpus. To address this gap, it is crucial to explore the capability of retrieval models to effectively handle the dynamic retrieval scenario inherent in KILTs.

In this work, we first introduce the continual document learning (CDL) task for KILTs and build a novel benchmark dataset named KILT++ based on the original KILT dataset for evaluation. Then, we conduct a comprehensive study of the use of pre-trained CorpusBrain on KILT++. Unlike the promising results in the stationary scenario, CorpusBrain is prone to catastrophic forgetting in the dynamic scenario, hence hampering retrieval performance. To alleviate this issue, we propose CorpusBrain++, a continual generative pre-training framework that enhances the original model along two key dimensions: (i) We employ a backbone-adapter

Research conducted when Ruqing Zhang was at the University of Amsterdam.

This work was funded by the Strategic Priority Research Program of the CAS under Grants No. XDB0680102, the National Natural Science Foundation of China (NSFC) under Grants No. 62472408, 62372431 and 62441229, the National Key Research and Development Program of China under Grants No. 2023YFA1011602, the Youth Innovation Promotion Association CAS under Grants No. 2021100, the Lenovo-CAS Joint Lab Youth Scientist Project, and the Strategic Priority Research Program of the CAS under Grants No. XDB0680301. This work was also (partially) funded by the Dutch Research Council (NWO), under project numbers 024.004.022, NWA.1389.20.183, and KICH3.LTP.20.006, and the European Union's Horizon Europe program under grant agreement No. 101070212. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

Authors' Contact Information: Jiafeng Guo, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China and University of Chinese Academy of Sciences, Beijing, China; e-mail: guojiafeng@ict.ac.cn; Changjiang Zhou, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China and University of Chinese Academy of Sciences, Beijing, China; e-mail: zhouchangjiang23s@ict.ac.cn; Ruqing Zhang (corresponding author), Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China and University of Chinese Academy of Sciences, Beijing, China; e-mail: zhangruqing@ict.ac.cn; Jiangui Chen, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China and University of Chinese Academy of Sciences, Beijing, China; e-mail: chenjiangui18z@ict.ac.cn; Maarten de Rijke, University of Amsterdam, Amsterdam, The Netherlands; e-mail: m.derijke@uva.nl; Yixing Fan, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China and University of Chinese Academy of Sciences, Beijing, China; e-mail: fanyixing@ict.ac.cn; Xueqi Cheng, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China and University of Chinese Academy of Sciences, Beijing, China; e-mail: cxq@ict.ac.cn.

ACM 1558-2868/2025/10-ART5
https://doi.org/10.1145/3763233

architecture: the dynamic adapter is learned for each downstream KILT task via task-specific pre-training objectives; the backbone parameters that are task-shared are kept unchanged to offer foundational retrieval capacity. (ii) We use an experience replay strategy based on exemplar documents that are similar to new documents, to prevent catastrophic forgetting of old documents. Empirical results demonstrate the effectiveness and efficiency of CorpusBrain++ in comparison to both traditional and generative information retrieval methods.

CCS Concepts: • **Information systems** → **Retrieval models and ranking**;

Additional Key Words and Phrases: Generative retrieval, Continual learning, Knowledge-intensive language tasks

ACM Reference format:
Jiafeng Guo, Changjiang Zhou, Ruqing Zhang, Jiangui Chen, Maarten de Rijke, Yixing Fan, and Xueqi Cheng. 2025. CorpusBrain++: A Continual Generative Pre-Training Framework for Knowledge-Intensive Language Tasks. *ACM Trans. Inf. Syst.* 44, 1, Article 5 (October 2025), 35 pages.
https://doi.org/10.1145/3763233

1 Introduction

Knowledge-intensive language tasks (KILTs) refer to a series of language-related tasks that require access to external knowledge sources such as Wikipedia for accurate answer generation [41]. In current mainstream approaches, a two-step process is commonly employed [5, 29, 62], consisting of a retriever and a reader. The retriever aims to retrieve relevant documents from large, external knowledge sources, while the reader is meant to synthesize the retrieved information to generate accurate and correct answers to the initial query. Thanks to the emergence of large-scale pre-trained generative language models [30, 42], the reader component has seen remarkable advances recently. The retriever component has primarily leaned on conventional discriminative methods [20], failing to fully capitalize on the potential advantages offered by generative models.

Generative retrieval (GR) has recently been proposed as an alternative retrieval paradigm [36]. In GR, the retrieval process is formalized as a **sequence-to-sequence (Seq2Seq)** learning problem, i.e., directly establishing a mapping from a query to its relevant document identifiers (docids). In essence, a single generative model is used to encode all information about the corpus into model parameters, allowing for end-to-end optimization and facilitating the alleviation of computational costs. As a result, GR stands out as a highly promising paradigm for retrieval in KILTs when compared to traditional discriminative methods. Specifically, previous research has investigated direct applications of pre-trained generative language models in the **natural language processing (NLP)** field, such as BART [30] and T5 [42], to the KILT retrieval task [4, 8, 55]. This approach involves initializing the model parameters with pre-trained generative models and subsequently fine-tuning them using golden query-docid pairs in downstream KILTs, which has demonstrated notable performance improvements in retrieval tasks.

Beyond the direct application of existing pre-trained generative models designed for NLP, there have been several pioneering studies on constructing generative pre-training tasks tailored for the KILT retrieval task. The underlying hypothesis is that using pre-training tasks that more closely resemble the relevance relationship between queries and documents in downstream KILT tasks can yield better retrieval performance [18, 26, 65]. A recent and representative contribution following this research domain pertains to CorpusBrain [9], whose results reported on the KILT leaderboard[1] showcase new state-of-the-art performance, surpassing strong baselines. The key idea of CorpusBrain is to construct pre-training data consisting of positive pairs of queries and docids

[1]https://eval.ai/challenge/689/leaderboard.

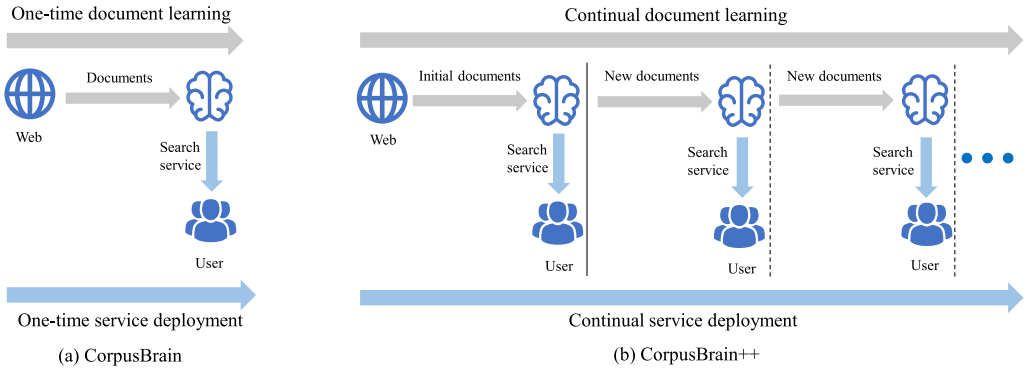


Fig. 1. Comparison of CorpusBrain and CorpusBrain++. CorpusBrain can solely support one-time document learning and service deployment, without the ability to assimilate new documents and dynamically update the knowledge base. Going beyond CorpusBrain, the dynamic CorpusBrain++ can support CDL and service deployment to adapt to evolving corpus in the realistic scenario. CDL, continual document learning.

that encompass various semantic granularities in downstream tasks. Subsequently, a transformer-based [58] encoder–decoder architecture is pre-trained by maximizing the likelihood of the output sequence with a standard Seq2Seq objective.

The majority of prior retrieval models developed for KILTs, including CorpusBrain, have primarily focused on a scenario with stationary knowledge sources, as shown in Figure 1(a): whenever they have finished learning, they remain unchanged when used in practice. In contrast to this static assumption, the accrual of knowledge over time is a ubiquitous phenomenon in most real-world scenarios, giving rise to new documents added to the underlying knowledge source. For instance, Wikipedia has experienced exponential growth in the number of documents[2] since its inception in 2001 [1], and new entities emerge in Wikipedia following, in many cases, the news cycle [19]. Therefore, to ensure that generalist information access systems based on a retriever-reader approach remains up-to-date and well-informed in the face of this ever-changing information landscape, it is imperative that they consistently expand their knowledge coverage. In traditional dense retrieval methods [25, 66], the process of incorporating new documents into the retrieval system is relatively straightforward; the encoded representations of the incremental documents can be directly added to an explicit external index, without requiring updates to the retrieval model itself. However, in the case of the state-of-the-art CorpusBrain model, the dynamic retrieval scenario poses a more significant challenge, mainly due to the use of an implicit parameterized index. Hence, it is of critical importance to investigate the ability of CorpusBrain to continuously accommodate the inclusion of new documents.

In this work, as illustrated in Figure 1(b), we make the first attempt to address the dynamic retrieval scenario for KILTs. We formally define the **continual document learning (CDL)** task for KILTs and outline the corresponding evaluation metrics. To facilitate fair and quantitative comparisons between different models w.r.t. their ability to tackle the CDL task, we introduce a novel benchmark dataset named KILT++, which is constructed by splitting the original KILT dataset [41] into distinct sessions to simulate the continual addition of new documents. Subsequently, we assess the performance of two relatively straightforward variants of the off-the-shelf pre-trained CorpusBrain model on the newly constructed KILT++ dataset, i.e., the direct insertion approach and the sequential pre-training approach. Our empirical findings confirm that, unlike the promising

[2]https://en.wikipedia.org/wiki/Wikipedia:Size_of_Wikipedia.

results achieved in the static scenario, CorpusBrain is vulnerable to catastrophic forgetting and inadequate in the ability to effectively and efficiently address the dynamic retrieval scenario.

To tackle the non-trivial CDL task, we propose a continual generative pre-training framework for KILTs, namely, CorpusBrain++ ("CorpusBrain + new documents"), to adapt CorpusBrain to the dynamic nature of constantly evolving corpora. CorpusBrain++ is designed to accurately retrieve both old and new documents for queries, without catastrophic forgetting of previous knowledge. To achieve this objective, we need to address two main challenges:

(1) How to expeditiously learn specific retrieval capacity for each KILT task as the corpus constantly evolves?
(2) How to prevent catastrophic forgetting the retrieval capacity already learned?

Specifically, we advance beyond the original CorpusBrain model in two key directions to solve the aforementioned challenges:

(1) In CorpusBrain++, we use a backbone-adapter architecture, wherein a dedicated adapter is employed for each downstream task to allow for capturing task-specific characteristics. The fixed backbone component serves as long-term memory to retain fundamental retrieval capacity, while the dynamic adapter component serves as short-term memory to rapidly learn incremental documents. To enable continual pre-training of the task-specific adapters, we design a pre-training task specifically tailored for each individual task.
(2) To avoid catastrophic forgetting of old documents, we use experience replay based on exemplar documents. We revisit old documents that are semantically similar to the incremental documents and apply the specific pre-training tasks for both the newly-arrived documents and the revisited ones.

We assess the performance of CorpusBrain++ on the constructed KILT++ dataset. Our empirical results demonstrate that CorpusBrain++ excels in efficiently and effectively handling the CDL task for KILTs. Further ablation studies are conducted, revealing the effectiveness of each individual component within the CorpusBrain++ architecture. Moreover, through our experimental analysis, we confirm that CorpusBrain++ successfully mitigates the occurrence of catastrophic forgetting of previously encountered documents and showcases the capability of positive forward knowledge transfer. Finally, we analyze the effectiveness-efficiency tradeoff of CorpusBrain++ and present a case study to further clarify its inner workings.

2 CDL for KILTs

Here, we first introduce the CDL task for KILTs and then describe the constructed KILT++ benchmark dataset, and finally elucidate the corresponding evaluation metrics to assess the CDL task for KILTs.

2.1 Task Formulation

Assume we have a large-scale base document set (i.e., Wikipedia articles) $\mathcal{D}_0$ and sufficiently many labeled query-document pairs $\mathcal{R}_0$ in downstream KILT tasks. Here, $\mathcal{R}_0$ contains all the labeled datasets in different KILT tasks, specifically including fact checking, entity linking, slot filling, open-domain **question answering (QA)**, and dialogue.

In the CDL task for KILTs, we assume there exist T batches of new documents $\{\mathcal{D}_1, \ldots, \mathcal{D}_t, \ldots, \mathcal{D}_T\}$, which arrive in a sequential manner as the time session grows. In any session $t \geq 1$, the corresponding labeled KILT data $\mathcal{R}_t$ is not available, i.e., $\mathcal{D}_t$ is only composed of newly encountered documents $\{d_t^1, d_t^2, \ldots\}$ without labeled queries relevant to these documents. Let the retrieval model after the tth update be $\mathcal{M}_t$ and the model parameters be Θ_t. For session t, the training

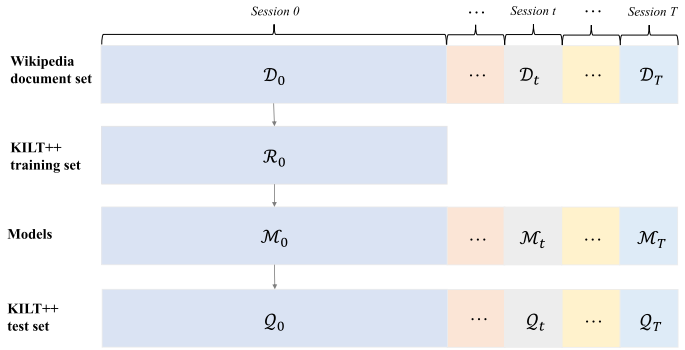


Fig. 2. Evaluation criteria of the CDL task for KILTs.

objective of CDL for KILT can be defined as updating Θ_{t-1} to Θ_t via the new document set $\mathcal{D}_t$ and previous datasets $\{\mathcal{D}_0, \ldots, \mathcal{D}_{t-1}\}$, such that $\mathcal{M}_t$ can simultaneously retrieve relevant documents from previously and newly arrived documents $\{\mathcal{D}_0, \ldots, \mathcal{D}_t\}$. To assess the retrieval performance of $\mathcal{M}_t$, we employ the test set $\mathcal{Q}_i^\delta, i \leq t$, where δ denotes the specific downstream KILT dataset, and i means that all relevant documents belong to $\{\mathcal{D}_0, \ldots, \mathcal{D}_i\}$. The evaluation criteria of the CDL task for KILTs is depicted in Figure 2.

2.2 Benchmark Construction

In order to study and evaluate the CDL task for KILTs, we build a new benchmark dataset based on the original KILT dataset [41], i.e., KILT++. The KILT dataset encompasses 11 datasets spanning 5 KILTs, which are all rooted in a shared knowledge source derived from a common Wikipedia snapshot. We split the datasets in each task into $T + 1$ sessions, i.e., session $0, \ldots, T$, to simulate a dynamic retrieval scenario for KILTs. We construct the benchmark dataset via the training set and the dev set since the KILT leaderboard imposes restrictions on the frequency of the submission for test performance.

To mimic the new arrival of documents, we set T to 4 and construct KILT++ as follows: (i) We randomly sample 60% documents from the full Wikipedia knowledge source to constitute the base document set $\mathcal{D}_0$. Then, we randomly divide the remaining Wikipedia documents into four incremental sets with the same number of documents, to serve as $\mathcal{D}_1, \ldots, \mathcal{D}_4$. (ii) To construct the labeled query-document pairs $\mathcal{R}_0$ corresponding to $\mathcal{D}_0$, we filter the original KILT training set by retaining only those query-document pairs where all relevant articles in the corresponding provenance exclusively belong to $\mathcal{D}_0$. To ensure the simulation of newly arrived documents, duplicate documents containing identical information to existing ones are systematically filtered out. Specifically, documents are considered duplicates if their similarity rate exceeds 80%. Refer to Section 5 for calculation of similarity rates between documents. (iii) To construct the test sets $\mathcal{Q}_0, \ldots, \mathcal{Q}_4$ corresponding to $\mathcal{D}_0, \ldots, \mathcal{D}_4$, we employ an iterative algorithm as follows. Initially, we filter the original KILT dev set by retaining only those query-document pairs where all relevant articles in the corresponding provenance exclusively belong to $\mathcal{D}_0$ and denote the constructed dataset as $\mathcal{Q}_0$. As for constructing $\mathcal{Q}_i, i \geq 1$, we iteratively filter the remaining KILT dev set by retaining only those query-document pairs where all relevant articles in the corresponding provenance exclusively belong to $\{\mathcal{D}_0, \ldots, \mathcal{D}_i\}$ and denote the constructed dataset as $\mathcal{Q}_i$.

It is worth noting that since the original KILT dataset inherently consists of 11 datasets spanning 5 downstream tasks, the derived $\mathcal{R}_0$ and $\mathcal{Q}_{(\cdot)}$ are essentially the same. We typically use $\mathcal{R}_0$ or $\mathcal{Q}_{(\cdot)}$ to denote the KILT++ training or test set as a whole, and when we would like to elucidate a specific

Table 1. Overall Statistics of Our Constructed KILT++ Benchmark Dataset

Dataset	Task	0		1	2	3	4
		#Train	#Test	#Test	#Test	#Test	#Test
FEV	Fact Checking	73,078	6,064	1,007	1,088	1,161	1,124
AY2	Entity Linking	10,745	2,847	553	418	441	525
WnWi	Entity Linking	-	2,052	360	332	288	364
WnCw	Entity Linking	-	3,381	652	460	605	501
T-REx	Slot Filling	1,304,897	2,876	488	504	553	579
zsRE	Slot Filling	94,980	2,226	353	379	417	349
NQ	Open-Domain QA	57,238	1,353	297	359	396	432
HoPo	Open-Domain QA	44,897	1,994	742	835	959	1,070
TQA	Open-Domain QA	33,152	2,457	599	732	708	863
ELI5	Open-Domain QA	-	745	172	199	201	190
WoW	Dialogue	39,823	2,162	391	131	199	171

KILT dataset such as FEV, we can add superscripts as a differentiation, i.e., $\mathcal{R}_0^{FEV}$ or $Q_{(\cdot)}^{FEV}$. Table 1 shows the overall statistics of our KILT++ benchmark dataset.

Furthermore, it is important to note that the random partitioning strategy used to divide the Wikipedia knowledge source treats all documents as equivalent, overlooking the temporal order of documents. Despite this disadvantage, we still use this strategy due to the following reasons: (i) In the original Wikipedia knowledge source for the KILT dataset, the initial timestamp indicating when a document was added is not available. Instead, only the timestamp of the most recent revision can be obtained. However, due to frequent updates, this revision timestamp no longer accurately reflects the original appearance of the document. Although it is theoretically possible to manually retrieve the original timestamps, for instance, by employing a web crawler, this approach would be highly resource-intensive and time-consuming, particularly given that our Wikipedia knowledge source comprises over 5.9 million documents. (ii) The size of the Wikipedia knowledge source varies across different years [39], resulting in fluctuations in the number of newly arrived documents each year. This variability may lead to an imbalance in the test set size. In contrast, the random partitioning strategy can help mitigate this issue. (iii) The random partitioning strategy is also used in existing work that explores **continual learning (CL)** for GR [6, 35]. Nevertheless, we acknowledge that the random partitioning approach does not reflect real-world temporal dynamics due to overlooking the temporal order of documents. This limitation may hinder the assessment of models' effectiveness in recognizing novel topics and adapting to temporal changes.

2.3 Evaluation Metrics

In this section, we group the evaluation metrics into three parts. First, we describe the metrics employed for assessing individual downstream datasets. Subsequently, we elucidate the metrics used for assessing individual downstream tasks. Finally, we provide the metrics employed for assessing all downstream datasets.

2.3.1 Assessing Individual Downstream Datasets. As illustrated in Table 1, for each session i, we have 11 specific KILT++ test sets denoted as Q_i^{δ} where δ denotes the specific downstream dataset such as FEV and AY2, whose relevant documents belong to $\{\mathcal{D}_0, \ldots, \mathcal{D}_i\}$. Suppose the performance

of the retrieval model $\mathcal{M}_t$ evaluated on the held-out test set Q_i^δ is $\mathcal{P}_{t,i}^\delta$:

$$\mathcal{P}_{t,i}^\delta = \sum_{q \in Q_i^\delta, d_q \in \{\mathcal{D}_0, \ldots, \mathcal{D}_i\}} g(d_q, \mathcal{M}_t(q)), i \leq t, \tag{1}$$

where d_q denotes the relevant document to the query $q \in Q_i^\delta$, and $g(\cdot)$ denotes a widely used evaluation metric for **information retrieval (IR)** such as recall [51]. Details of the evaluation metrics used will be described in Section 5.3.

When it comes to individually assessing M_t on a specific downstream dataset δ, we compare the vertical performance VP_t^δ of different approaches on Q_t^δ in the same session t:

$$VP_t^\delta = \mathcal{P}_{t,t}^\delta. \tag{2}$$

2.3.2 Assessing Individual Downstream Tasks. When it comes to individually assessing M_t on a specific downstream task τ in session t, we take the average vertical performance across all specific datasets under this task as the metric:

$$VP_t^\tau = \frac{1}{|D^\tau|} \sum_{\delta \in D^\tau} VP_t^\delta, \tag{3}$$

where D^τ denotes the set of all specific downstream datasets that belong to the task τ.

2.3.3 Assessing All Downstream Datasets. To give a comprehensive retrieval performance across all downstream datasets in the session t, we employ the vertical performance VP_t:

$$VP_t = \frac{1}{|D|} \sum_{\delta \in D} VP_t^\delta, \tag{4}$$

where D denotes the set of all specific downstream datasets.

Since we pay more attention to the comprehensive retrieval capability across all downstream datasets and tasks, we merely employ the following across-all-session metrics to assess the trends in comprehensive retrieval performance. Hence, we first define the comprehensive retrieval performance of the model on the test set Q_i after the learning phase of session t $\mathcal{P}_{t,i}$ to facilitate the elaboration of the later-defined metrics:

$$\mathcal{P}_{t,i} = \frac{1}{|D|} \sum_{\delta \in D} \mathcal{P}_{t,i}^\delta, \tag{5}$$

where D denotes the set of all specific downstream datasets. To provide a metric for assessing all downstream datasets across all sessions, following [33, 35], we also employ the following evaluation metrics:

(1) Average Performance (AP) is used to measure the average performance at the conclusion of training with the entire existing data sequence:

$$AP = \frac{1}{T+1} \sum_{i=0}^{T} \mathcal{P}_{T,i}, \tag{6}$$

where to the term $\mathcal{P}_{T,i}$ represents the retrieval performance of the model on the test set Q_i after training on all available data up to session T. By averaging these performance scores over all sessions (from 0 to T), AP can reflect the model's average retrieval ability across all sessions.

(2) Backward Transfer (BWT) is used to evaluate the effect of learning a new session on the performance of all previous sessions:

$$BWT = \frac{1}{T}\sum_{i=0}^{T-1} \max_{t\in\{0,\ldots,T-1\}} (\mathcal{P}_{t,i} - \mathcal{P}_{T,i}), \tag{7}$$

where the term $\mathcal{P}_{t,i}$ indicates the comprehensive retrieval performance of the model on the test set $\mathcal{Q}_i$ after the learning phase of session t. The expression $\mathcal{P}_{t,i} - \mathcal{P}_{T,i}$ quantitatively represents the discrepancy in performance on $\mathcal{Q}_i$ between session t and T. By taking the maximum over all sessions $t \in \{0, \ldots, T-1\}$, the most significant backward transfer effect can be identified. The average of these maximum values across all sessions provides a comprehensive measure of how well the model retains knowledge from earlier sessions when new sessions are introduced.

(3) Forward Transfer (FWT) is used to measure the ability to learn when confronted with a new session:

$$FWT = \frac{1}{T}\sum_{t=1}^{T} \mathcal{P}_{t,t}, \tag{8}$$

where the term $\mathcal{P}_{t,t}$ represents the comprehensive retrieval performance of the model on the test set $\mathcal{Q}_i$ after the learning phase of session t. By averaging these performance scores from session 1 to T, we assess the effectiveness of the model in using prior knowledge to enhance learning in new sessions.

3 Analysis of CorpusBrain on CDL

In this section, based on our constructed KILT++ benchmark dataset, we conduct an empirical analysis of CorpusBrain to investigate its performance on CDL tasks.

3.1 Background

Our method is designed based on CorpusBrain [9], and thus we would like first to provide a brief overview of CorpusBrain before delving into the details of our proposed extension. CorpusBrain is a pre-trained GR model for KILTs, exhibiting state-of-the-art retrieval performance for KILTs.

3.1.1 Model Architecture. In CorpusBrain, a transformer-based [58] encoder–decoder architecture is used to capture the relevance between queries and docids, which incorporates an encoder to yield the query representation and a decoder to generate the relevant docids. In the implementation of CorpusBrain, the titles of Wikipedia pages are selected as docids.

3.1.2 Pre-Training Tasks. Three self-supervised pre-training tasks are devised to generate pseudo-query-docid pairs from documents and hence facilitate retrieval for KILTs. The pre-training tasks in CorpusBrain are carefully designed based on a prevailing hypothesis that using pre-training tasks that bears greater resemblance to downstream tasks results in superior fine-tuning effectiveness. Specifically, three pre-training tasks with different granularity are introduced:

— **Inner sentence selection (ISS)**. Inner sentences are randomly sampled from the document as pseudo-queries, with the document and destination pages linked by anchor texts serving as relevant target documents. The ISS task is designed to capture sentence-level semantic context.

— **Lead paragraph selection (LPS)**. Leading paragraphs are drawn from the document as pseudo-queries, with document and destination pages linked by anchor texts serving as

relevant target documents as well. The LPS task allows for capturing paragraph-level semantic information.

— **Hyperlink identifier prediction (HIP)**. The corresponding sentences of randomly sampled anchors, along with the surrounding contextual sentences, are chosen as pseudo-queries, with the destination pages linked by the anchors serving as the relevant target documents. The LPS task is used to capture inter-document semantic relevance.

3.1.3 Pre-Training Process. In CorpusBrain, the "pre-train and fine-tune" paradigm is employed to adapt to multiple downstream KILT tasks. In the pre-training phase, a checkpoint of BART [30] is first applied to initialize the parameters to reduce the cost of training from scratch. After generating pairs of pseudo-queries and docids by the aforementioned pre-training tasks, a standard Seq2seq learning objective, i.e., maximum likelihood estimation [37], is employed to optimize the model, denoted as:

$$\mathcal{L} = \sum_{q \in f(\mathcal{D})} \sum_{m} \sum_{n} \log p(w_{m,n} \mid w_{\leq m, <n}, q; \Theta), \tag{9}$$

where $\mathcal{D}$ represents the knowledge source corpus, $f(\cdot)$ signifies the transformation function of pre-training tasks, q refers to the constructed pseudo-query, $w_{m,n}$ denotes the nth token in the mth docid related to q, and Θ refers to the model parameters.

3.1.4 Fine-Tuning Process. To further adapt to multiple downstream KILT tasks, the model is then fine-tuned on all KILT training datasets across five tasks through a multi-task training objective. By applying the fine-tuned model to the KILT test set and decoding with a constrained beam search strategy on the docid prefix tree, as illustrated in the KILT leaderboard, CorpusBrain can achieve the top performance on a number of downstream tasks.

3.2 Overall Performance

As for directly using the pre-trained CorpusBrain model to address the CDL task for KILTs, we build two naive CorpusBrain variants as follows.

— *The direct insertion approach* (denoted as *Direct*), inserts new docids, i.e., Wikipedia titles, from the incremental corpus $\mathcal{D}_t$ directly into the docid prefix tree, without updating the backbone parameter Θ_0.

— *The sequential pre-training approach* (denoted as *Sequential*), sequentially pre-trains the model via $\mathcal{D}_0 \sim \mathcal{D}_t$ with self-supervised pre-training tasks.

Experimental Results. Based on the performance reported in Table 2, the following observations can be made: (i) The *Direct* method suffers from a significant drop of approximately 20% in terms of VP in the first incremental session, compared to BM25 with only 9%, which demonstrates that the *Direct* method cannot learn incremental documents well when they arrive. Moreover, the *Direct* method solely depends on the generalization capability of the backbone model, without a mechanism to learn knowledge within new documents. (ii) The *Sequential* variant exhibits a more considerable drop in terms of VP in the first incremental session, quantitatively about 47%. We can also observe a high BWT score, which demonstrates the *Sequential* variant is prone to catastrophic forgetting. Notably, the *Sequential* method even demonstrates inferior performance compared to the traditional BM25 in terms of *VP* in session 4. (iii) On the whole, neither of the two naive variants can effectively tackle the CDL task for KILTs.

Based on our analysis, we conclude that the naive variants of CorpusBrain are either deficient in their ability of forward transfer or prone to catastrophic forgetting. Hence, the CDL task for KILTs poses a non-trivial challenge for CorpusBrain.

Table 2. Performance of Naive CorpusBrain Variants on the CDL Task

Model	Q_0	Q_1	Q_2	Q_3	Q_4	**AP↑**	**BWT↓**	**FWT↑**
BM25	27.61	25.11	22.97	23.84	22.92	22.26	2.78	23.71
Direct	59.72	47.86	44.87	46.21	45.42	48.28	0.67	46.09
Sequential	59.72	31.84	27.09	26.81	22.48	17.67	19.89	27.05

We evaluate the retrieval performance on Q_i in terms of VP. As for AP, BWT, and FWT, ↑ indicates that higher is better and ↓ that lower is better.

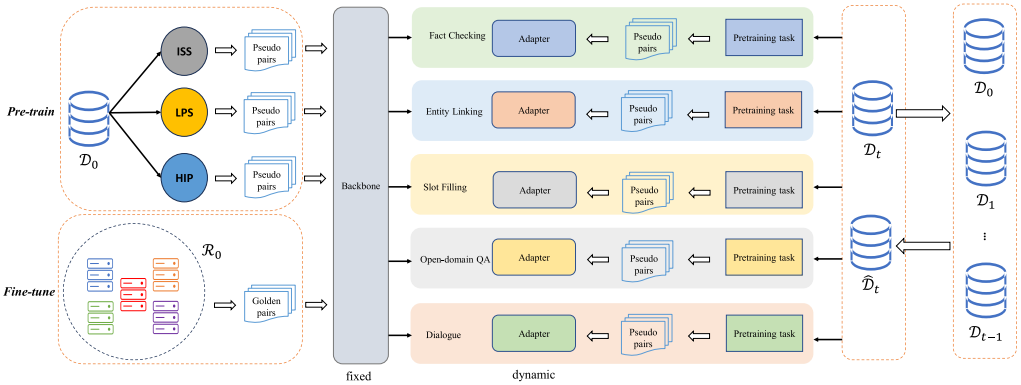


Fig. 3. Illustration of our proposed CorpusBrain++ method. The backbone first involves pre-training on the initial base document set D_0 through the ISS, LPS, and HIP pre-training tasks, and fine-tuning with golden pairs derived from the KILT++ training set. To accommodate each task, a specific adapter is allocated, and a dedicated pre-training task is introduced to mimic the characteristics of downstream input queries. In addition to the incremental document set D_t, we also revisit semantically similar documents to D_t from previous sessions, thereby generating pseudo pairs and continually pre-training the adapters.

4 Methodology

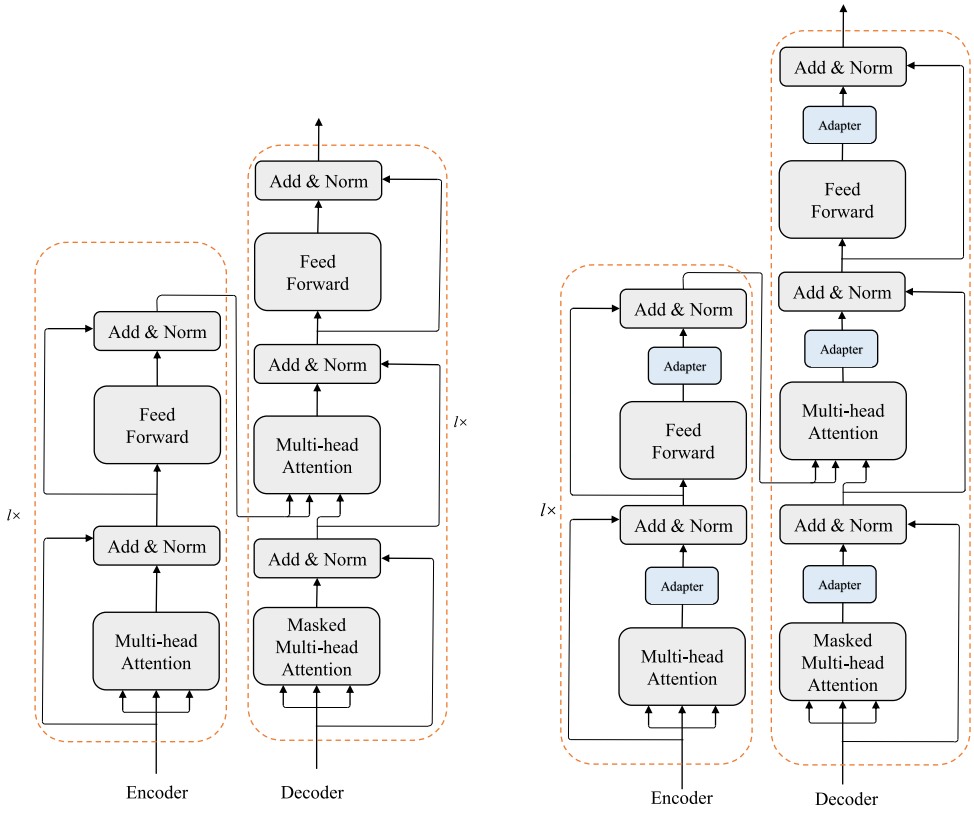
In this section, we introduce a novel continual generative pre-training framework for KILTs, i.e., CorpusBrain++. We first introduce our model design, and then describe the technical details. Finally, we explain the learning and inference processes of the model.

4.1 Model Overview

According to our analysis in Section 3, naive variants of CorpusBrain struggle to address the CDL challenge effectively and efficiently. We attribute this to the fact that the characteristics of CL for KILTs are neglected. Since KILTs encompass multiple downstream tasks with distinct forms of input queries, modeling each task separately could facilitate CL.

Inspired by the parameter-isolation paradigm in the CL field [34, 48], we propose a continual generative pre-training framework to incrementally learn new documents for KILTs, namely CorpusBrain++. As shown in Figure 3, CorpusBrain++ incorporates three key features:

(1) First, we employ a backbone-adapter architecture. The shared backbone undergoes pre-training on $\mathcal{D}_0$ and subsequent fine-tuning on $\mathcal{R}_0$, with the parameters held fixed to ensure consistent provision of task-shared knowledge. To accommodate the evolving corpus, a specific adapter is used for each KILT task to efficiently learn new documents, which aims to retain the characteristics of specific data in each task.



(a) Architecture of CorpusBrain (b) Architecture of CorpusBrain++

Fig. 4. The architectures of CorpusBrain and CorpusBrain++.

(2) Second, we design a specific pre-training objective for each KILT task, to resemble the relevant relationship between queries and documents in each specific task. In this way, we can continually pre-train the task-specific adapters.
(3) Third, to avoid catastrophic forgetting, we revisit some old documents and apply the specific pre-training tasks for both new and old documents for continual pre-training.

4.2 Shared Backbone

Here, we present the structure of the shared backbone, which constitutes a transformer-based encoder–decoder architecture. As depicted in Figure 4(a), both the encoder and decoder are generally composed of l layers. Each layer of the encoder comprises multiple components including a multi-head self-attention sub-layer (*MHSA*), a feed-forward neural network sub-layer (*FFN*), and a residual connection subsequently followed by layer normalization (*RCLN*). Likewise, each layer of the decoder consists of multiple components incorporating a masked multi-head self-attention sub-layer (*MMHSA*), a multi-head cross-attention sub-layer (*MHCA*), a feed-forward neural network sub-layer (*FFN*), and a residual connection subsequently followed by layer normalization (*RCLN*).

Next, we introduce the basic multi-head attention mechanism, then specify three sub-layers incorporating the attention mechanism and introduce the principle of the feed-forward layer.

Multi-Head Attention Mechanism. Given the input hidden state $h^q, h^k, h^v \in \mathbb{R}^{n\times d}$, the ith attention head can be formulated as:

$$\text{Attention}_i(\mathbf{h}^q, \mathbf{h}^k, \mathbf{h}^v) = \sum_m \text{softmax}\left(\frac{W_i^q \mathbf{h}^q \cdot W_i^k \mathbf{h}^k}{\sqrt{d/m}}\right) W_i^v \mathbf{h}^v, \tag{10}$$

where $W_i^{(\cdot)} \in \mathbb{R}^{d/m\times m}$ are trainable projection matrices. Additionally, the outputs of multiple attention heads are fed into a multi-head attention layer, of which the mechanism can be formulated as follows:

$$MH(\mathbf{h}^q, \mathbf{h}^k, \mathbf{h}^v) = \text{Concat}(\text{Attention}_1(\mathbf{h}^q, \mathbf{h}^k, \mathbf{h}^v), \ldots, \text{Attention}_n(\mathbf{h}^q, \mathbf{h}^k, \mathbf{h}^v))W^o, \tag{11}$$

where $W^o \in \mathbb{R}^{d\times d}$ is the learned transformation matrix.

In the self-attention layers, i.e., *MHSA* and *MMHSA*, $\mathbf{h}^q$, $\mathbf{h}^k$, and $\mathbf{h}^v$ all refer to the input hidden state $\mathbf{h}$. An attention mask is employed in *MMHSA* to preserve the auto-regressive property. In the cross-attention layers, i.e., *MHCA*, $\mathbf{h}^q$ comes from the previous decoder layer while $\mathbf{h}^k$ and $\mathbf{h}^v$ come from the output of the encoder layer.

Feed-Forward Sub-Layer. The feed-forward sub-layer is a position-wise fully connected FFN, which can be formulated as follows:

$$FFN(\mathbf{h}) = \sigma(\mathbf{h}W_1 + b_1)W_2 + b_2, \tag{12}$$

where $W_1 \in \mathbb{R}^{d\times 4d}$ and $W_2 \in \mathbb{R}^{4d\times d}$ are trainable transformation matrices, $b_1 \in \mathbb{R}^{4d}$ and $b_2 \in \mathbb{R}^d$ are trainable bias terms.

Each sub-layer, denoted as $S \in \{MHSA, MMHSA, MHCA, FFN\}$, uses a residual connection followed by layer normalization (*RCLN*), which can be formulated as follows:

$$RCLN(\mathbf{h}) = LN(S(\mathbf{h}) + \mathbf{h}), \tag{13}$$

where $LN(\cdot)$ denotes layer normalization.

4.3 Task-Specific Adapter

As mentioned before, we assign a task-specific adapter for each downstream KILT task and continually pre-train the adapters to accomplish the CDL task for KILTs. Below, we provide the technical details for adapter insertion and adapter structure.

4.3.1 Adapter Insertion. To capture a specific query-to-docid mapping for each task, we maintain an adapter module for each task independently, i.e., the task-specific adapter. The adapter is a series of compact and efficient modules inserted after the sub-layers of the transformer, defined as:

$$\text{Adapter}(\mathbf{h}) = g(\mathbf{h}W_{down})W_{up}, \tag{14}$$

where $W_{down} \in \mathbb{R}^{d\times r}$ and $W_{up} \in \mathbb{R}^{r\times d}$ are trainable down-projection and up-projection matrices correspondingly, and g denotes the transformation function.

As depicted in Figure 4(b), we insert adapters into both the encoder and decoder modules. The underlying reasons are two-fold: (i) The input queries vary significantly across distinct downstream tasks, including semantic granularity and formats, hence we insert adapters into the encoder module. (ii) The query-docid mapping also varies across downstream tasks, therefore we insert adapters into the decoder module apart from the encoder module.

Given the promising results achieved in [23, 53], as depicted in Figure 4(b), we insert adapters in an inside way. In other words, we place an adapter behind each sub-layer $S \in \{MHSA, MMHSA,$

MHCA, *FFN*}. The formulation of each sub-layer in Equation (13) undergoes the following transformation:

$$\text{Inside}(\mathbf{h}) = \text{LN}(\text{Adapter}(S(\mathbf{h})) + \mathbf{h}). \tag{15}$$

4.3.2 Adapter Structure. In the area of multi-task learning and transfer learning [23], previous work has made much progress on the issue of how to design more efficient transformation functions g in Equation (14). We employ a simple yet effective adapter structure named the low-rank layer, which exhibits promising empirical results in [7, 52]. In this adapter structure, the g function is defined as a low-rank transformation, i.e., an identity function.

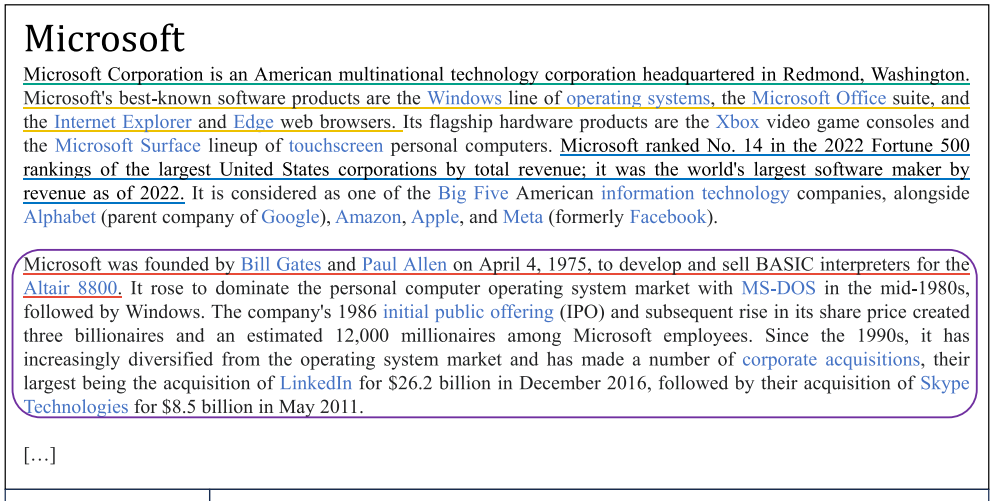
4.4 Task-Specific Pre-Training Objective

To facilitate continually pre-training the task-specific adapters, we carefully design a specific pre-training objective for each downstream KILT task. The principle of each pre-training task is to mimic the relevant relationship between queries and documents in the corresponding downstream task as much as possible. The objective of each pre-training task mainly incorporates two components, i.e., input pseudo-queries and output docids. In terms of distinct downstream KILT tasks, distinctions of input pseudo-queries primarily manifest themselves in semantic granularity. During the construction of input pseudo-queries, we first build preliminary pseudo-queries based on semantic granularity, and further refine them to accommodate the characteristics of downstream KILT tasks. On the other hand, distinctions of output docids primarily manifest themselves in the number of supporting documents. During the construction of output docids, we determine the number of output docids for each downstream task according to the specific output scenario associated with the corresponding task.

4.4.1 Fact Checking. In terms of fact checking, the input pseudo-queries and output docids are constructed as follows:

— *Input pseudo-queries*. The construction of input pseudo-queries includes the following steps:
  (1) From the perspective of semantic granularity, fact-checking queries typically encompass sentence-level semantic context. For instance, a typical query example that *Windows are the software products of Microsoft* is at the sentence level. Hence we use the ISS pre-training task to sample sentences from the document. Specifically, given a document d, we randomly draw l inner sentences from d to form the preliminary pseudo-queries.
  (2) We can also observe that fact-checking queries are typically short in length (seven words in this case) and entity-centric (*Windows* and *Microsoft* in this case). To further mimic the aforementioned characteristics, we randomly sample an n-gram span from each preliminary pseudo-query.
  (3) The document title is regarded as the core entity within d in previous work [13, 60]. Therefore, We add the document title to the beginning of the sampled span to construct the final pseudo-query. An example input pseudo-query for fact checking is given in Figure 5(i).

— *Output docids*. From the perspective of target document numbers, fact checking might require multiple supporting documents to judge the authenticity of a given claim. For example, we might need two supporting documents in this case, i.e., a document titled *Windows* and another titled *Microsoft*. Following ISS, we first randomly sample o anchor texts within d. Apart from d, we also treat the destination pages linked by these o anchors as the relevant documents. As depicted in Figure 5(i), we concatenate the docid of d and the docids of these o relevant documents with a separator `[SEP]`. By this means the final output sequence could be constructed, which allows for dynamic predictions of relevant documents.



(i) Fact checking	**Input:** Microsoft's best-known software products are the Windows line of operating systems. **Output:** Microsoft [SEP] Microsoft Office
(ii) Entity linking	**Input:** Microsoft was founded by [START_ENT] Bill Gates [END_ENT] and Paul Allen on April 4, 1975, to develop and sell BASIC interpreters for the Altair 8800. **Output:** Bill Gates
(iii) Slot filling	**Input:** Microsoft [SEP] headquarters location **Output:** Microsoft
(iv) Open-domain QA	**Input:** What are Microsoft rankings of the largest United States corporations by total revenue? **Output:** Microsoft [SEP] Amazon
(v) Dialogue	**Input:** Microsoft was founded […] $8.5 billion in May 2011. When did Microsoft dominate the personal computer operating system market with MS-DOS? **Output:** Microsoft [SEP] initial public offering

Fig. 5. Illustration of specific pre-training tasks for each KILT task. Anchor texts are marked in blue. The colored underlines in Wikipedia content correspond to the source text of the corresponding KILT task in the table below. Input and output refer to the constructed input pseudo-queries and corresponding output docids.

4.4.2 Entity Linking. In terms of entity linking, the input pseudo-queries and output docids are constructed as follows:

— *Input pseudo-queries.* The construction of input pseudo-queries includes the following steps:

(1) From the perspective of semantic granularity, entity-linking queries typically involve inter-document semantic relations. We provide a typical entity-linking query as an example: `[START_ENT]` *Bill Gates* `[END_ENT]` *is best known as the co-founder of Microsoft Corporation, one of the world's largest and most successful technology companies.* This query carries the inter-document semantic relation between the document titled *Bill Gates* and the document titled *Microsoft*. Based on the above analysis, we employ a variant of the HIP pre-training task to construct preliminary pseudo-queries. Specifically, given a document d, we randomly select l anchor texts within d. Subsequently, we locate the corresponding sentences containing the selected anchor texts as the preliminary pseudo-queries.

(2) As illustrated in Figure 5(ii), entity-linking queries typically contain special tokens to indicate the entity boundary, i.e., `[START_ENT]` and `[END_ENT]`. To further mimic this characteristic, we insert special tokens revealing entity boundaries as well. Specifically, for each preliminary pseudo-query, we insert `[START_ENT]` and `[END_ENT]` to the left

and right of the anchor text. An example input pseudo-query for entity linking is given in Figure 5(ii).

— *Output docids*. From the perspective of target document numbers, only a unique target docid is required for entity linking (*Bill Gates* in this case). To this end, for each pseudo-query, the output docid refers to the docid of the destination page linked by the selected anchor text. As illustrated in Figure 5(ii), the constructed output docid refers to *Bill Gates* in this case.

4.4.3 Slot Filling. In terms of slot filling, the input pseudo-queries and output docids are constructed as follows:

— *Input pseudo-queries*. The construction of input pseudo-queries includes the following steps:
(1) From the perspective of semantic granularity, slot-filling queries typically involve sentence-level semantic context. For example, a typical slot-filling query *Microsoft* `[SEP]` *headquarters location* mainly contains sentence-level semantics. Therefore, we first use the ISS pre-training task to sample sentences from the document. Given a document d, we randomly draw l inner sentences from d to form the preliminary pseudo-queries.
(2) Each slot-filling query typically constitutes a subject entity (*Microsoft* in this case) and a relational predicate (*headquarters location* in this case) pre-defined in the candidate set. To closely mimic this characteristic, we first train a relation detector by fine-tuning the BERT model [15] on the slot-filling training set in the initial session, i.e., $\mathcal{R}_0^{T-REx}$ and $\mathcal{R}_0^{zsRE}$. The relation detector allows for the prediction of relation types within each sampled sentence. We retain the top-k predicted relation types for each sampled sentence.
(3) We treat the core entity within d, i.e., the document title, as the subject entity. Subsequently, for each sampled sentence, we separately concatenate the subject entity with the top-k predicted relation types using a separator (`[SEP]`) to form k final pseudo-queries. An example input pseudo-query for slot filling is given in Figure 5(iii).

— *Output docids*. From the perspective of target document numbers, the slot-filling task tends to require a single supporting document (*Microsoft* in this case). Therefore, as depicted in Figure 5(iii), the output docid refers to the docid of d for each pseudo-query.

4.4.4 Open-Domain QA. In terms of open-domain QA, the input pseudo-queries and output docids are constructed as follows:

— *Input pseudo-queries*. The construction of input pseudo-queries includes the following steps:
(1) In the case of open-domain QA, queries typically involve sentence-level semantic context. Therefore, we first use the ISS pre-training task to sample sentences from the document. Given a document d, we randomly draw l inner sentences from d to form the preliminary pseudo-queries.
(2) As illustrated in Figure 5(iv), queries for open-domain QA are typically in the form of questions. Additionally, we can observe that QA queries are often short in length (10 words in this case). Moreover, the queries could sometimes incorporate an entity to locate the supporting documents (*Microsoft* in this example). To further mimic the aforementioned characteristics, for each preliminary pseudo-query, we randomly sample an n-gram span. Subsequently, for each sampled span, we randomly select an interrogative word from a pre-defined candidate set, and then add it to the beginning of the span.
(3) We insert the document title between the interrogative word and the sampled span to emphasize the core entity within d. An example input pseudo-query for open-domain QA is given in Figure 5(iv).

— *Output docids*. From the perspective of target document numbers, more than one supporting document might be required to accomplish the QA task. Following ISS, we first randomly

sample o anchor texts within d. Subsequently, we treat the destination pages linked by these o anchors as the relevant documents. Finally, as shown in Figure 5(iv), we concatenate the docid of d and the docids of the o relevant documents with a separator [SEP]. By this means the final output sequence could be constructed, which allows for dynamic predictions of relevant documents.

4.4.5 Dialogue. In terms of dialogue, the input pseudo-queries and output docids are constructed as follows:

— *Input pseudo-queries*. The construction of input pseudo-queries includes the following steps:
(1) From the perspective of semantic granularity, queries in the dialogue typically involve paragraph-level semantic information. Hence, we first apply the LPS pre-training task to sample paragraphs from the document. Given a document d, we sample the leading l paragraphs from d to form the preliminary pseudo-queries.
(2) As illustrated in Figure 5(v), dialogue queries tend to comprise a long conversation context and a question related to the context. To further mimic this characteristic, we treat each preliminary pseudo-query as a conversation context and construct a question for each preliminary pseudo-query. For each preliminary pseudo-query, we randomly sample an n-gram span. Subsequently, for each sampled span, we randomly select an interrogative word from a pre-defined candidate set, and then add it to the beginning of the span. We insert the document title between the interrogative word and the sampled span to emphasize the core entity within d. By this means we could construct l paragraph-level questions.
(3) Instead of replacing the preliminary pseudo-queries with the constructed questions, we concatenate the preliminary pseudo-queries and the constructed questions to yield the final pseudo-queries. An example input pseudo-query for dialogue is given in Figure 5(v).

— *Output docids*. From the perspective of target document numbers, multiple supporting documents might be needed to comprehend the conversation context and provide a correct answer. Following LPS, we first randomly sample o anchor texts within d. Subsequently, we treat the destination pages linked by these o anchors as the relevant documents. As shown in Figure 5(v), we concatenate the docid of d and the docids of the o relevant documents with a separator [SEP]. By this means the final output sequence could be constructed, which allows for dynamic predictions of relevant documents.

4.5 Learning Process

Here, we detail the learning process deployed in our method. First, we elucidate the learning process of newly arrived documents. We then explain the strategy of rehearsing old documents, which serves as a countermeasure against catastrophic forgetting. Lastly, we present the overall learning objective.

4.5.1 Learning New Documents. In the learning process, we continually learn new documents by updating the parameters of task-specific adapters. When confronted with a new document set $D_t, t \geq 1$, we first generate pairs of pseudo-queries and docids by applying the task-specific pre-training tasks to each incremental document. For each downstream task, we can learn new documents by continually pre-training the corresponding task-specific adapter with the constructed pairs. The learning objective is detailed in Section 4.5.3.

4.5.2 Rehearsing Old Documents. To avoid catastrophic forgetting of old documents while learning new documents, we perform experience rehearsal of old documents. When learning new knowledge or skills, humans tend to draw upon similar past experiences as a reliance [10, 11, 22].

Inspired by this fact about human cognition, we employ a cluster-based strategy for rehearsing old documents in our framework, which allows for the consideration of semantic similarity. For each new session $t \geq 1$, we first cluster old documents in previous sessions into k categories. In our framework, we employ the K-means clustering algorithm [21] to cluster documents in $\mathcal{D}_0, \ldots, \mathcal{D}_{t-1}$ into k categories. In terms of representing each document, we directly use the docid to facilitate efficiency. We feed all docids of the old documents into the K-means clustering algorithm:

$$\{C_1, \ldots, C_k\} = \text{K-means}\left(\left\{\gamma \mid \gamma \in \bigcup_{i=0}^{t-1} \mathcal{D}_i\right\}\right), \tag{16}$$

where γ denotes the docid, and $C_{(\cdot)}$ refers to the document cluster.

When a stream of documents $\mathcal{D}_t$ arrives, for each new document $d_t^i \in \mathcal{D}_t$, we first judge the specific cluster to which the document belongs. Subsequently, we randomly select n old documents from the corresponding cluster. After repeating the aforementioned process for each $d_t^i \in \mathcal{D}_t$, we can construct an experience set $\hat{\mathcal{D}}_t$, which share semantic similarity with $\mathcal{D}_t$. Similar to learning new documents, we use the aforementioned pre-training tasks on $\hat{\mathcal{D}}_t$ to generate pairs of pseudo-queries and docids. For each downstream task, we can review old documents by continually pre-training the corresponding task-specific adapter with the constructed pairs. The learning objective is detailed in Section 4.5.3.

4.5.3 Overall Learning Objective. For each new session t, the parameters of each task-specific adapter are independently updated, while the parameters of the shared backbone are kept unchanged. First, we initialize the parameters of each adapter by inheriting from the previous session. Under exceptional circumstances wherein $t = 1$, the parameters of task-specific adapters are randomly initialized since it is the first incremental session. Subsequently, we construct pairs of pseudo-queries and docids for both the new document set $\mathcal{D}_t$ and the old document set $\hat{\mathcal{D}}_t$ following Section 4.5.1 and Section 4.5.2. Finally, for each downstream KILT task, we separately update the parameters of the task-specific adapter with the generated pairs tailored for the corresponding task. We apply a standard Seq2Seq learning objective to continually pre-train the task-specific adapters:

$$\mathcal{L} = \sum_{\tau \in \mathcal{T}} \sum_{(q,\gamma) \in f^\tau(\mathcal{D}_t \cup \hat{\mathcal{D}}_t)} \log p(\gamma, \Theta_t^\tau \mid q; \Theta_{t-1}^\tau), \tag{17}$$

where $\mathcal{T}$ denotes the task set, τ represents a specific downstream task in $\mathcal{T}$, $f^\tau(\cdot)$ refers to the transformation function of the pre-training task dedicated for τ, (q, γ) denotes the constructed pairs of pseudo-queries and docids. $\mathcal{D}_t$ refers to the new document set, and $\hat{\mathcal{D}}_t$ refers to the old document set derived from the cluster-based strategy of document rehearsal. Θ_{t-1}^τ represents the meta parameters of the task-specific adapter dedicated for τ before the tth update, and Θ_t^τ refers to the meta parameters of the task-specific adapter dedicated for τ after the tth update.

4.5.4 Discussion on the Frozen Backbone. In our framework, we keep the parameters of the backbone frozen to update the GR model with reduced computational overhead. This design choice is justified based on the following two key assumptions: (i) The model is assumed to have fully acquired fundamental retrieval capabilities by digesting the knowledge embedded in the initial document collection. (ii) The introduction of newly arrived documents does not induce extreme **out-of-distribution (OOD)** issues. We acknowledge that updating the parameters of the backbone becomes necessary under the following circumstances: (i) The initial document collection contains insufficient knowledge to equip the model with foundational retrieval capabilities, for example because of limited domain coverage or insufficient data quantity. (ii) The arrival of new documents leads to the emergence of a substantial number of new topics, concepts, or thematic

clusters, hence invalidating the retrieval capabilities of the backbone. It is important to note that, quantitatively, if retrieval performance degrades over time despite repeated adapter retraining, it may become necessary to retrain the backbone. Despite the limitations of the frozen backbone, parameter-efficient fine-tuning can offer several additional advantages: (i) It can inherently mitigate catastrophic forgetting of existing knowledge by maintaining similarity to the original GR model. (ii) It enables the use of task-specific adapters instead of individual model copies.

4.6 Inference Process

During the inference phase, when confronted with the test set $Q_t, t \geq 1$, we initially categorize the test data into distinct task groups following the corresponding relationship illustrated in Table 1. Subsequently, for the test data of each task, we activate both the backbone model and the corresponding task-specific adapter to generate target docids incorporating both task-shared and task-specific knowledge. Additionally, we use a constrained beam search approach [2, 13] to confine the generated docids within a pre-defined set of docids. It is noteworthy that, in session t, only docids corresponding to existing documents, namely, $\mathcal{D}_0, \ldots, \mathcal{D}_t$, are incorporated into the prefix tree. Since docids refer to document titles in this work, we confine the output sequence within the constraints of a document title prefix tree. Specifically, each node in the prefix tree corresponds to a token. When traversing from the root to a specific node, all nodes on the path collectively constitute a document title.

5 Experimental Settings

In this section, we explain our experimental settings.

5.1 Models

5.1.1 Baselines. In this study, we conduct a comparative analysis involving our proposed CorpusBrain++, traditional IR models, and generative IR models.

— *Traditional IR models.* (i) *BM25* [47] is a typical sparse retrieval model, which uses term-based features to model the relevance between queries and documents. (ii) *DocT5Query* [38] expands each document in the corpus with pseudo-queries generated by a fine-tuned T5 [42] model. (iii) *DPR* [25] is a representative dense retrieval model, which models the semantic relevance between queries and documents via a dual-encoder architecture. (iv) *ANCE* [61] selects hard training negatives globally from the entire corpus with an asynchronously updated ANN index to facilitate the contrastive learning of dense retrieval methods. For the traditional IR models, our empirical results encompass both incremental and non-incremental scenarios thanks to the high reproducibility provided by Pyserini[3] and Tevatron.[4]

— *Generative IR models.* We first consider several generative IR models in stationary scenarios, including (i) *GENRE* [13], which directly fine-tunes BART via multi-task training on the labeled KILT training datasets and supervised BLINK datasets [60]; and (ii) *SEAL* [4], which applies a BART-based autoregressive search engine to generate distinctive n-grams as docids. Given that the focus of both models is confined to the non-incremental scenario, we only compare them with CorpusBrain++ in terms of non-incremental retrieval performance. It is worth noting that CorpusBrain++ degenerates to CorpusBrain in the non-incremental scenario. Furthermore, we explore some advanced generative IR models fitting in dynamic scenarios, including (i) *DSI++*, which continually fine-tunes DSI over new documents and allocates a

[3] https://github.com/castorini/pyserini.
[4] https://github.com/texttron/tevatron.

unique integer as the docid for each new document; and (ii) *CLEVER*, which introduces a technique named incremental product quantization to assign a docid to each new document.

5.1.2 Ablation Models. Apart from the naive *Direct* and *Sequential* variants, whose modifications on top of CorpusBrain have been introduced in Section 2, we modify CorpusBrain++ from three perspectives to explore the effectiveness of each component:

— *The impact of different model architectures*. We assess two variants of CorpusBrain++, which continually pre-train the backbone rather than the adapter with the task-specific pre-training objective: (i) For CorpusBrain++$_{-Adapter(ST)}$ we continually pre-train the backbone in a single-task manner, i.e., we continually pre-train the backbone independently for each downstream task with the corresponding task-specific pre-training objective. And (ii) for CorpusBrain++$_{-Adapter(MT)}$ we continually pre-train the backbone in a multi-task manner, i.e., we continually pre-train the backbone jointly for all downstream tasks with the proposed task-specific pre-training objective.

— *The impact of different pre-training objectives*. We design a variant CorpusBrain++$_{OriPT}$, where we continually pre-train the task-specific adapter with the original ISS, LPS, and HIP pre-training tasks.

— *Analyze the impact of different document rehearsal strategies*. We implement two variants: (i) CorpusBrain++$_{Random}$, where we randomly sample some old documents to construct query-docid pairs via the task-specific objective; and (ii) CorpusBrain++$_{-Rehearsal}$, where we eliminate the strategy of old document rehearsal, i.e., we solely construct query-docid pairs with incremental documents.

5.2 Implementation Details

In this work, we use the Wikipedia document title as the docid for simplicity. Following Chen et al. [9], we employ $BART_{large}$ as the backbone architecture. We use $BERT_{base}$ to encode Wikipedia document titles and use the encoding representation of the special token [CLS] as the representation of each Wikipedia document. The similarity rate between documents is calculated by normalizing the cosine similarity of their representations to a range of [0, 1]. We can divide the learning process into two phases, i.e., the initial phase and the incremental phase.

In the initial phase, we incorporate the pre-training and fine-tuning process following Chen et al. [9]. For the pre-training process, we use three pre-training tasks originally defined in [9], i.e., ISS, LPS, and HIP, to construct pre-training data on $\mathcal{D}_0$, and the number of constructed pseudo-pairs is determined the same as [9]. Additionally, we employ a learning rate of $3e^{-5}$ alongside the Adam optimizer [27], incorporating a warmup technique with a warmup ratio of 0.1. Moreover, we set the weight decay to 0.01, set the label smoothing to 0.1, set the gradient norm clipping to 0.1, and set the batch size to 8,192 tokens. In terms of the fine-tuning process, we fine-tune the backbone model via multi-task training on $\mathcal{R}_0$ spanning 5 distinct downstream tasks following [4, 9, 13], it is worth noting, as illustrated in Table 1, that not all 11 datasets in $\mathcal{R}_0$ have an accessible training set. We set the learning rate to $3e^{-5}$ and set the batch size to 4,096 tokens.

In the incremental phase, i.e., in session t, $t \geq 1$, we first construct pre-training data by the well-designed task-specific pre-training tasks with the newly arrived documents $\mathcal{D}_t$ and the revisited documents $\hat{\mathcal{D}}_t$, and then continually pre-train the task-specific adapters via the constructed pre-training data. For the task-specific adapters, we set the hidden size to 1,024 and set the reduction factor to 4. For the task-specific pre-training tasks, we select distinct hyper-parameters according to the characteristics of different downstream tasks. Following Chen et al. [9], for all task-specific pre-training tasks except entity linking and slot filling, o is in [0, 1, 2, 3, 4] with a probability of [70%, 20%, 5%, 3%, 2%], respectively. For fact checking, we set l to 3 and n to 10. For entity linking,

we set l to the maximum number of anchors within d to explore inter-document relations as much as possible as anchors linked to other Wikipedia pages are relatively limited. For slot filling, we set l to 3 and k to 1. For open-domain QA, we set l to 3 and n to 10. For dialogue, we set l to 1 and n to 10. When it comes to rehearsing old documents, we use K-means clustering algorithm to cluster the encoding representation of document titles. We set k to 1,024 and set the maximum number of iterations in the K-means clustering algorithm to 20. In the continual pre-training phase, we set the learning rate to $1e^{-5}$ a warmup technique with a warmup ratio of 0.1.

At inference time, we use constrained beam search with 10 beams and set the maximum decoding steps to 15. As for the entity linking sub-task, we limit the input sequence to a maximum of 384 tokens by truncating either the left, right, or both parts of the context surrounding an entity mention.

In the non-incremental scenario, all models, except for sparse retrievers, are trained using the original Wikipedia knowledge source and the full KILT training set. Sparse retrievers, including BM25 and DocT5Query, construct their indices directly from the original Wikipedia knowledge source. For the initial session of the incremental scenario, the experimental setup mirrors that of the non-incremental scenario: all models, with the exception of sparse retrievers, are trained on $\mathcal{D}_0$ and $\mathcal{R}_0$, while sparse retrievers, including BM25 and DocT5Query, construct their indices directly from $\mathcal{D}_0$. For session t ($t \geq 1$) in the incremental scenario, sparse retrievers reconstruct their indices using the Wikipedia knowledge source available up to session t, i.e., $\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_t$. Dense retrievers generate dense embeddings for the documents in $\mathcal{D}_t$ and incorporate them into the existing dense index. Generative retrievers are continually trained on $\mathcal{D}_t$ to facilitate CDL.

As for DSI++, we reimplement it following the empirical settings specified in the original publications since the source code has not yet been released. As for CLEVER, we borrow the original code and adapt it to the scenario of multiple downstream KILT tasks. To facilitate a fair comparison, we adopt the $BART_{large}$ model architecture for both models consistent with CorpusBrain++ in our implementation.

5.3 Evaluation Metrics

In order to assess the retrieval performance of distinct models on the KILT++ test set, as recommended in the official instructions of KILT [41], we adopt R-precision (%) as the specific evaluation metric, i.e., the $g(\cdot)$ function in Equation (1). R-precision is defined as follows:

$$R\text{-}precision = \frac{r}{R}, \tag{18}$$

where R refers to the number of Wikipedia pages in the golden provenance set, and r denotes the number of relevant documents present within the top-R retrieved pages. We report page-level R-precision in our analysis.

6 Experimental Results

In this section, we compare CorpusBrain++, baselines and variants in the dynamic retrieval scenario. The aim is to assess the effectiveness of CorpusBrain++ in comparison to existing baselines and ablation variants. We then evaluate CorpusBrain++ from a task-specific perspective, considering the retrieval performance in distinct tasks. We also evaluate the catastrophic forgetting phenomenon and forward transfer capability of distinct models. Additionally, we analyze the effectiveness-efficiency tradeoff. Finally, we present a case study to provide a specific illustration of the effectiveness of our proposed method.

6.1 Baseline Comparison

CorpusBrain has been shown to outperform traditional IR methods such as BM25 and DPR as well as generative IR methods such as GENRE and SEAL in the non-incremental retrieval scenario [9].

Table 3. Non-Incremental Retrieval Performance of Baseline Models on Full KILT Dev Sets

Model	FEV	AY2	WnWi	WnCw	T-REx	zsRE	NQ	HoPo	TQA	ELI5	WoW	Avg.
BM25	26.0	3.41	0.09	3.23	49.4	51.53	25.31	37.71	26.53	5.97	18.83	22.55
DocT5Query	41.62	3.01	0.15	2.59	40.76	50.21	31.30	36.96	38.78	7.5	23.67	25.14
DPR	54.33	4.62	0.44	0.82	16.82	35.61	52.1	29.42	67.29	15.46	37.95	28.62
ANCE	62.53	3.97	0.29	0.41	19.5	39.15	55.30	28.29	59.96	13.87	38.47	29.25
SEAL	67.8	-	-	-	58.9	78.8	43.6	**54.3**	41.8	-	36.0	-
GENRE	84.68	92.75	87.69	70.57	79.68	94.84	64.26	51.82	71.11	13.47	56.32	69.74
CorpusBrain	**85.03**	**92.86**	**88.64**	**71.35**	**80.22**	**98.49**	**64.61**	52.23	**71.71**	**14.33**	**59.72**	**70.84**

The performance is evaluated in terms of the R-precision metric. **Bold** indicates the best performance.

To streamline our work, we do not repeat the experiments and refer, instead, to [9] for more empirical details. In this work, we solely evaluate the retrieval performance of distinct models in the incremental scenario. In the following, we first analyze the incremental effectiveness of distinct models on 11 individual downstream datasets. Additionally, we analyze the incremental effectiveness of distinct models on five individual downstream tasks, each of which contains one to four datasets. Finally, we analyze the overall incremental effectiveness of distinct models on all downstream datasets.

6.1.1 Non-Incremental Performance on Full Data. Table 3 exhibits the non-incremental retrieval performance of baseline models on the full KILT datasets. As demonstrated in Table 3, CorpusBrain outperforms the baseline methods, encompassing sparse, dense, and generative retrievers, by achieving superior performance on 10 out of 11 datasets and demonstrating the highest average retrieval performance. This observation underscores CorpusBrain's robust retrieval capability in non-incremental scenarios.

6.1.2 Incremental Performance of Individual Downstream Datasets. Table 4 provides an overview of the incremental retrieval performance of different models on specific KILT datasets. When we look at the retrieval results presented in Table 4, we observe the following:

(1) Sparse retrieval methods including BM25 and DocT5Query demonstrate close retrieval capability across various distinct downstream datasets, and so do dense retrieval methods including DPR and ANCE. Dense retrievers generally exhibit better performance than sparse models in the majority of downstream datasets, and the underlying reason may be that the supervised dense retrievers can learn more semantic characteristics of downstream datasets than unsupervised sparse retrievers. However, the comparison results on the T-REx and zsRE datasets are opposite, which may be due to the important differences between the input query format of these two datasets (phrases) and the query format learned by DPR (sentences).
(2) GR baselines, i.e., DSI++ and CLEVER, exhibit retrieval ability to some degree in non-incremental scenarios, especially on the AY2 and NQ datasets. Nevertheless, this retrieval ability fails to be effectively sustained in incremental scenarios. This may be attributed to the exclusive focus of these models on homogeneous downstream queries during CL, while the dynamic retrieval scenario for KILTs is dominated by heterogeneous downstream queries. The failure of previous GR models highlights that it is a non-trivial challenge to continually model the relevant relationship between heterogeneous downstream queries and the corresponding documents.
(3) In general, CorpusBrain++ consistently surpasses traditional retrieval methods and GR models in the majority of downstream datasets.

Table 4. Dynamic Retrieval Performance on Individual Downstream KILT++ Datasets

Session	Model	FEV	AY2	WnWi	WnCw	T-REx	zsRE	NQ	HoPo	TQA	ELI5	WoW
0	BM25	32.58	4.00	8.77	4.14	56.82	60.51	30.75	**42.50**	32.52	6.98	24.10
	DocT5Query	50.85	3.9	1.17	3.05	49.24	59.61	38.51	41.47	45.91	9.66	27.15
	DPR	49.96	2.95	0.97	0.30	12.00	23.67	45.23	24.90	46.56	**15.57**	27.98
	ANCE	57.47	5.02	0.39	1.01	18.78	36.34	46.39	30.64	48.57	**15.57**	29.64
	DSI++	6.96	56.06	9.84	22.86	0.07	0.58	22.32	1.58	25.27	2.95	12.72
	CLEVER	15.5	67.93	17.93	28.54	1.91	6.42	39.1	4.36	38.26	4.97	19.29
	CorpusBrain++	**85.57**	**83.32**	**54.24**	**53.95**	**84.11**	**97.53**	**51.22**	40.52	**58.85**	9.4	**38.16**
1	BM25	31.17	5.79	1.11	3.22	53.69	58.64	27.61	41.11	27.55	8.14	18.16
	DocT5Query	43.66	3.25	0.83	1.38	43.24	30.99	28.96	37.47	38.73	12.79	25.58
	DPR	48.46	2.89	1.11	0.77	11.07	24.08	**40.40**	24.12	46.41	12.79	27.88
	ANCE	53.84	2.17	0.83	0.15	15.78	35.69	39.64	29.99	47.77	**16.28**	38.34
	DSI++	0.15	0.72	0.0	0.46	0.2	0.0	1.35	0.07	4.67	1.16	0.26
	CLEVER	0.55	0.18	0.0	0.15	0.82	0.0	4.04	0.2	1.34	0.0	7.93
	Direct	76.03	**52.26**	47.22	34.05	77.66	94.90	24.92	37.87	45.74	4.65	31.20
	Sequential	68.45	8.50	3.89	6.60	70.29	77.90	19.87	34.91	39.07	12.79	7.93
	CorpusBrain++	**77.14**	49.37	**59.72**	**36.2**	**78.07**	**96.03**	25.93	**41.44**	**48.58**	4.65	**39.39**
2	BM25	28.84	0.48	0.30	4.35	50.79	56.99	28.69	39.10	27.87	4.52	10.69
	DocT5Query	40.49	0.48	0.0	4.13	41.07	53.83	33.43	37.31	39.34	4.52	**13.74**
	DPR	44.36	2.15	1.81	0.87	7.74	20.58	43.18	24.73	44.95	**14.57**	3.05
	ANCE	51.49	5.02	0.3	0.22	15.28	34.3	**49.86**	30.72	48.06	**14.57**	5.34
	DSI++	0.37	0.0	0.6	0.22	0.0	0.0	3.34	0.06	2.73	0.5	0.0
	CLEVER	1.52	0.0	0.0	0.0	0.6	0.0	1.67	0.6	0.27	0.5	0.0
	Direct	75.70	47.37	43.07	35.43	74.01	96.04	27.30	38.68	41.80	5.03	9.16
	Sequential	70.53	4.07	4.22	8.26	55.95	63.32	13.37	29.70	32.38	8.54	7.63
	CorpusBrain++	**80.64**	**49.28**	**62.05**	**38.48**	**74.21**	**96.83**	31.2	**43.65**	**48.63**	7.54	7.63
3	BM25	30.19	5.67	0.35	3.80	48.10	57.07	26.52	37.9	31.07	3.98	17.59
	DocT5Query	40.68	5.22	0.0	2.64	41.77	51.8	31.82	38.11	43.64	5.47	17.59
	DPR	38.39	5.67	0.69	0.0	11.21	21.34	38.64	22.89	47.18	12.94	**23.62**
	ANCE	46.53	6.8	1.04	0.0	14.1	33.33	**45.71**	27.95	48.7	**14.43**	21.61
	DSI++	0.43	0.0	0.0	0.33	0.0	0.0	1.77	0.05	1.69	1.0	0.0
	CLEVER	0.99	0.0	0.35	0.0	0.18	1.44	2.53	0.1	0.56	0.5	0.0
	Direct	64.15	56.92	44.44	38.51	74.50	95.20	28.03	35.71	45.76	8.96	16.08
	Sequential	57.04	9.52	3.82	16.36	47.02	62.35	9.60	27.22	34.46	8.46	19.10
	CorpusBrain++	**69.24**	**58.05**	**65.62**	**39.83**	**73.78**	**96.88**	29.8	**40.72**	**50.71**	6.47	19.6
4	BM25	27.74	1.90	0.0	2.00	44.73	46.99	25.93	38.60	30.24	9.47	24.56
	DocT5Query	40.45	2.48	0.0	2.0	38.0	47.56	30.32	37.43	41.02	7.89	24.56
	DPR	43.65	2.48	0.27	0.0	9.67	21.78	38.19	21.68	44.15	13.16	23.39
	ANCE	48.68	2.86	0.0	2.0	11.92	34.96	**46.99**	27.06	50.08	**16.32**	31.84
	DSI++	1.78	0.38	0.0	0.0	0.0	0.0	0.46	0.14	1.16	0.0	0.0
	CLEVER	1.01	0.0	0.0	0.0	0.17	0.86	1.85	0.09	0.12	0.0	0.0
	Direct	66.57	**53.14**	47.53	34.93	**66.67**	95.13	23.61	35.47	48.09	1.58	26.90
	Sequential	61.35	2.29	4.40	5.79	38.69	53.58	9.95	25.75	29.90	7.37	8.19
	CorpusBrain++	**71.98**	50.86	**67.03**	**39.72**	**66.67**	**97.13**	23.84	**40.47**	**53.42**	2.11	**32.16**

We evaluate the performance of $Q_0^\sigma, \ldots, Q_4^\sigma$ in terms of vertical performance (VP) with the R-precision metric. **Bold** indicates the best performance.

(4) CorpusBrain++ performs worse than DPR on the NQ dataset in incremental sessions, the underlying reason may be that NQ serves as one of the training datasets employed by DPR, rendering DPR a stronger retriever on the NQ dataset.
(5) CorpusBrain++ exhibits unstable retrieval performance on the AY2, ELI5 and WOW datasets. What these datasets have in common is that the input queries are all of a relatively long and complex form. For example, ELI5 is a dataset for long-form QA, which contains complex and

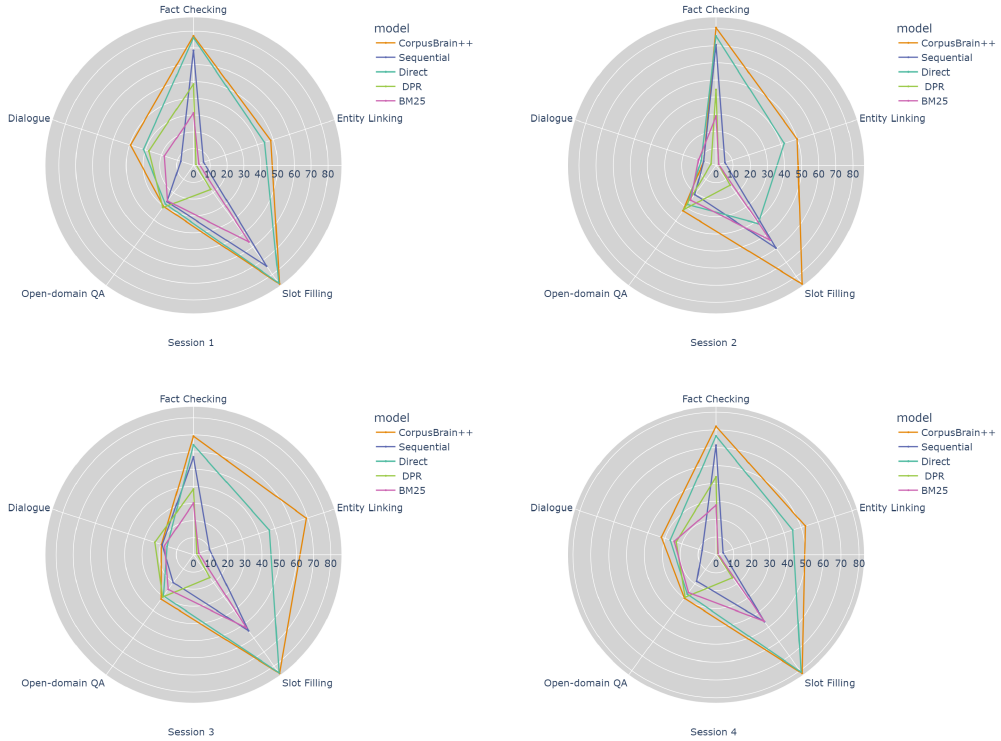


Fig. 6. Dynamic retrieval performance of distinct models on downstream KILT tasks. We evaluate the performance of $Q_0^\tau, \ldots, Q_4^\tau$ in terms of vertical performance (VP) with the R-precision metric.

diverse questions that require explanatory multi-sentence answers [16]. More challengingly, no training data is provided for ELI5. This phenomenon may suggest that CorpusBrain++ is unstable in retrieving results in the face of long and complex input queries, and a future avenue is to boost the query length and complexity in the task-specific pre-training objectives.

6.1.3 Incremental Performance of Individual Downstream Tasks. To facilitate the readability of the schematic representation, we select BM25, DPR, Direct, Sequential, and CorpusBrain++ to analyze the incremental performance of individual downstream tasks. As illustrated in Figure 6, CorpusBrain++ outperforms traditional retrieval methods and naive generative variants across nearly all downstream KILT tasks during all sessions. When we examine the enhancements achieved by CorpusBrain++ in each downstream task, we observe the following:

(1) The gain in dynamic retrieval performance is notably more significant in the fact checking and entity linking tasks. We attribute this to the fact that both of these tasks are entity-centric. Consequently, our model can effectively learn from new documents by emphasizing the entities within these documents as part of the task-specific pre-training objective.
(2) The gain in dynamic retrieval performance for the slot filling task, compared to other downstream tasks, is relatively modest. This phenomenon can likely be attributed to the fact that the pre-training tasks within CorpusBrain effectively align with the downstream format of the slot filling task during the initial session, allowing the backbone model to retain a substantial retrieval capability for this specific task.

Table 5. Comprehensive Retrieval Performance of Distinct Models on Q_i in Terms of Vertical Performance (VP) with the R-Precision Metric

Model	Q_0	Q_1	Q_2	Q_3	Q_4	AP↑	BWT↓	FWT↑
BM25	27.61	25.11	22.97	23.84	22.92	22.26	2.78	23.71
DocT5Query	30.05	24.26	24.39	25.34	24.70	24.71	2.86	24.67
DPR	22.74	21.82	18.91	20.23	19.86	18.49	2.77	20.21
ANCE	26.34	25.50	23.20	23.65	24.79	22.75	2.91	24.28
DSI++	14.66	0.82	0.71	0.48	0.36	1.4	2.72	0.59
CLEVER	22.20	1.38	0.47	0.6	0.38	0.62	5.49	0.7
Direct	59.72	47.86	44.87	46.21	45.42	48.28	0.67	46.09
Sequential	59.72	31.84	27.09	26.81	22.48	17.67	19.89	27.05
CorpusBrain++$_{-Adapter(ST)}$	59.72	26.15	24.66	23.23	22.26	22.76	10.55	24.08
CorpusBrain++$_{-Adapter(MT)}$	59.72	33.06	33.29	31.71	31.78	32.97	6.26	32.46
CorpusBrain++$_{OriPT}$	59.72	45.33	41.15	40.5	39.29	41.85	4.18	41.57
CorpusBrain++$_{Random}$	59.72	49.62	47.05	47.67	47.12	49.69	0.8	47.86
CorpusBrain++$_{-Rehearsal}$	59.72	49.26	47.56	47.85	46.96	49.55	1.05	47.91
CorpusBrain++	**59.72***	**50.59***	**49.10***	**50.06***	**49.58***	**52.01***	**0.41***	**49.83***

As for AP, BWT, and FWT, ↑ indicates that higher is better and ↓ that lower is better. **Bold** indicates the best performance. $*$ indicates statistically significant improvements over all baselines (p-value < 0.05).

(3) The improvement in dynamic retrieval performance exhibits a relatively unstable pattern in the open-domain QA task and the dialogue task. We attribute this variability to the nature of these two tasks, which are not entity-centric. In these tasks, it is not guaranteed that the entities within the target document will appear in the input query, and hence, the characteristics of downstream tasks are hard to simulate in the task-specific pre-training objective.

6.1.4 Incremental Performance of All Datasets. Table 5 presents an overview of the overall dynamic retrieval performance of different models.

Overall Analysis. Based on Table 5, we find that:

(1) Sparse retrievers such as BM25 and DocT5Query demonstrate a similar tendency as the knowledge source corpus evolves. Specifically, BM25 and DocT5Query suffer from a drop in terms of VP in the first incremental session, and the observation across later sessions is relatively consistent. Despite BWT suggests that the impact of CDL on backward transfer is relatively acceptable, both sparse retrievers exhibit inferior retrieval and forward transfer capabilities compared with the generative IR model CorpusBrain++.
(2) Dense retrievers including DPR and ANCE show relative stable performance across all sessions. Similarly, AP and FWT suggest that they also suffer from weaker retrieval and forward transfer capabilities in comparison to CorpusBrain++.
(3) Generative IR baselines, i.e., DSI++ and CLEVER, almost completely lose the retrieval capability during the incremental phase, the underlying reason may be two-fold. First, both baselines focus on the dynamic scenario of a single downstream task, while the dynamic retrieval scenario for KILTs comprises various downstream tasks spanning multiple semantic granularity. Second, Wikipedia titles, given their robust semantic structure, can effectively serve as docids in the retrieval scenario for KILTs, whereas the atomic integer docid employed in DSI++ and the product quantization code used in CLEVER may increase the difficulty of CL.

(4) In line with the analysis presented in Section 3, when naive variants of CorpusBrain, namely Direct and Sequential, are exposed to the arrival of new documents, we observe a substantial decline in retrieval performance, especially in the Sequential variant. This phenomenon serves as compelling evidence that off-the-shelf CorpusBrain is susceptible to catastrophic forgetting and faces challenges when adapting to the dynamic retrieval scenarios for KILTs.
(5) Taken as a whole, based on the data presented in Table 5, it is evident that CorpusBrain++ consistently achieves the best retrieval performance across all metrics, including VP, AP, BWT, and FWT. Given the superior performance of CorpusBrain++ when compared to traditional and GR methods, it is evident that our proposed CorpusBrain++ can effectively adapt to the dynamic retrieval scenario.

Impact of Different Model Architectures. When we compare variants with different model architectures, i.e., the backbone-only architecture incorporating CorpusBrain++$_{-Adapter(ST)}$ and CorpusBrain++$_{-Adapter(MT)}$, and the backbone-adapter architecture incorporating all other CorpusBrain++ variants, we can observe that:

(1) Variants using the backbone-adapter architecture consistently demonstrate superior retrieval performance across all metrics when compared to variants employing the backbone-only architecture. The backbone-adapter architecture exhibits significantly lower BWT scores, indicating its enhanced ability to mitigate catastrophic forgetting. This phenomenon can be attributed to the inherent characteristics of the backbone-adapter architecture, which permits updates solely to fractional meta-parameters, specifically the adapter parameters, while maintaining the stability of the backbone. This design choice ensures the preservation of fundamental retrieval capabilities within the backbone.
(2) The variant CorpusBrain++$_{-Adapter(ST)}$ is outperformed by CorpusBrain++$_{-Adapter(MT)}$. A potential explanation for this discrepancy lies in the training strategy. During the initial session, both pre-training and fine-tuning stages adhere to the multi-task training approach, creating a divergence from the single-task training methodology employed during the incremental session.

Impact of Different Pre-Training Objectives. When we look at variants with different pre-training objectives, i.e., CorpusBrain++$_{OriPT}$ which continually pre-trains the backbone-adapter architecture with the original pre-training tasks, and CorpusBrain++ which continually pre-trains the backbone-adapter architecture with our proposed task-specific pre-training objective, we can observe and analyze as follows:

(1) Despite retaining the backbone-adapter architecture to counteract catastrophic forgetting, CorpusBrain++$_{OriPT}$ exhibits retrieval performance even worse than the naive Direct variant when new documents arrive. This observation reveals that it is not feasible to directly employ general pre-training tasks to accommodate distinct downstream tasks in incremental sessions.
(2) Regarding the question of why the multi-task learning mechanism proves effective during the initial session but falters in incremental sessions, the underlying explanation may lie in the accessibility of golden query-docid pairs for downstream KILT tasks. During the initial session, the availability of these golden query-docid pairs minimizes the introduction of data noise, ensuring a more stable fine-tuning stage. Nevertheless, during incremental sessions, we encounter a significant challenge in the form of insufficiently labeled query-docid pairs. As a consequence, persisting with pre-training our model following the paradigm of multi-task learning results in the introduction of substantial levels of data noise. To address this challenge, we shift our approach to follow the work line of single-task learning, allowing for

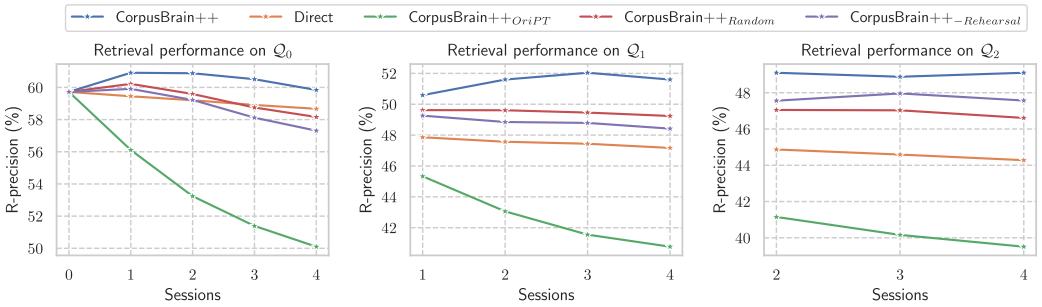


Fig. 7. The catastrophic forgetting phenomenon of different models. Based on the comprehensive retrieval performance of all datasets, we illustrate the retrieval performance on Q_i in terms of vertical performance (VP) with the R-precision metric.

a more focused learning objective. Thanks to the backbone-adapter architecture, assigning an adapter for each KILT task incurs minimal computational and storage overhead.

Impact of Different Document Rehearsal Strategies. When we focus on variants with distinct document rehearsal strategies, i.e., CorpusBrain++$_{-Rehearsal}$ without document rehearsal, CorpusBrain++$_{Random}$ with a random rehearsal strategy, and CorpusBrain++ with a document rehearsal strategy based on semantic similarity, we can observe that:

(1) Among the variants, CorpusBrain++$_{-Rehearsal}$ displays the poorest performance in terms of BWT, underscoring the efficacy of the old document rehearsal strategy in further alleviating the phenomenon of catastrophic forgetting.
(2) CorpusBrain++ demonstrates superior performance across all metrics when compared to CorpusBrain++$_{Random}$, highlighting the effectiveness of the semantic-similarity-based document rehearsal strategy.

6.2 Assessing Catastrophic Forgetting

In order to further assess the forgetting behavior of distinct models, we illustrate the forgetting curve of distinct models as the session grows in terms of the retrieval performance on Q_0, Q_1, and Q_2. We select models with a relatively low BWT score including CorpusBrain++, Direct, CorpusBrain++$_{OriPT}$, CorpusBrain++$_{Random}$, and CorpusBrain++$_{-Rehearsal}$ for comparison. In Figure 7 we observe that: (i) The forgetting curve for CorpusBrain++$_{OriPT}$ exhibits a notably steeper decrease compared to other models. The underlying reason might be that retrieval capabilities for downstream KILT tasks are significantly weakened in the incremental phase without the task-specific pre-training objective. (ii) Inconsistent with Q_0 and Q_1, CorpusBrain++$_{Random}$, surprisingly, even reinforces catastrophic forgetting in terms of retrieval performance on Q_2 compared with CorpusBrain++$_{-Rehearsal}$. This phenomenon may imply that an improper rehearsal strategy such as the random sampling strategy may even sometimes play a negative role in alleviating catastrophic forgetting. (iii) The forgetting curve for the naive variant Direct is relatively flat, which we attribute to the fact that the model parameters are constantly kept fixed in Direct. Influenced by the frozen parameters, we can observe that Direct exhibits relatively worse retrieval performance in the incremental phase compared with other models. (iv) CorpusBrain++ allows almost complete prevention of catastrophic forgetting, which proves the effectiveness of the task-specific pre-training objective and the semantic-similarity-based document rehearsal strategy.

Table 6. The Forward Transferring Phenomenon of Different Models

Model	Q_0	Q_1	Q_2	Q_3	Q_4
Individual	59.72	50.59	47.24	47.17	46.54
CorpusBrain++	**59.72***	**50.59***	**49.10***	**50.06***	**49.58***

Based on the comprehensive retrieval performance of all datasets, we illustrate the retrieval performance on Q_i in terms of vertical performance (VP) with the R-precision metric. **Bold** indicates the best performance.
$*$ indicates statistically significant improvements over all baselines (p-value < 0.05).

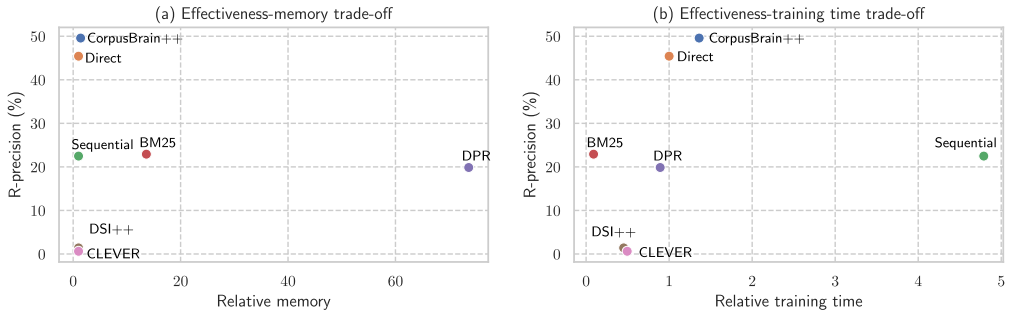


Fig. 8. Comparison on effectiveness-memory tradeoff and effectiveness-training time tradeoff. Up and left is better. Relative memory usage and the relative training time are with respect to Direct.

6.3 Assessing Forward Transfer

In order to further assess the forward transfer ability of CorpusBrain++, which measures the capacity to use prior knowledge in adapting to new sessions, we design a new variant denoted as Individual. In Individual, we continually pre-train the task-specific adapters individually with the tailored pre-training objective in each session, without initializing the parameters of adapters from the prior session. Importantly, the training process for both methods during session 0 and session 1 is entirely equivalent. As illustrated in Table 6, CorpusBrain++ consistently outperforms Individual by a substantial margin starting from session 2. This result further confirms the robust forward transfer ability of CorpusBrain++.

6.4 Effectiveness-Efficiency Tradeoff

We undertake a further comparison of the effectiveness-efficiency tradeoff across various models. Specifically, we select traditional IR methods including BM25 and DPR, as well as generative IR methods including DSI++, CLEVER, Direct, Sequential, and CorpusBrain++. As for effectiveness, we evaluate the retrieval performance on Q_T in terms of VP after finishing training for all T sessions. For the memory overhead, we calculate the disk space usage of each model after finishing document learning of all sessions. For the temporal overhead, we compare the total training time incurred at the conclusion of document learning. In the case of CorpusBrain++, the task-specific adapters are continually pre-trained in parallel, hence we only count the training time of the most time-consuming adapter in the incremental phase. As depicted in Figure 8, both memory and training time are presented as relative ratios with respect to Direct, which enhances the clarity of the comparison.

When we look at the effectiveness-memory tradeoff presented in Figure 8(a), we observe that: (i) The memory overhead of generative IR methods is significantly more modest than that of traditional IR methods, exhibiting a difference of an order of magnitude. This discrepancy can be attributed to the inherent characteristics of parameterized indexes employed in generative IR methods, which demonstrate higher rates of information compression when contrasted with the external indexes used in traditional IR methods. (ii) Compared to Direct, CorpusBrain++ demonstrates a significant enhancement in effectiveness while incurring only a marginal increase in storage overhead, which further demonstrates the effectiveness and efficiency of the backbone-adapter architecture. (iii) CorpusBrain++ achieves superior retrieval performance surpassing other models, while incurring only a slight increase in memory overhead compared to Direct.

When we look at the effectiveness-training time tradeoff presented in Figure 8(b), we observe that: (i) While the training process of traditional IR methods takes a shorter time than Direct, it is noteworthy that the retrieval effectiveness achieved is suboptimal. (ii) Sequential incurs a substantial training time cost, primarily attributable to the update of all backbone parameters. Despite this investment in training time, Sequential fails to deliver satisfactory retrieval performance. (iii) CorpusBrain++ demonstrates superior retrieval performance while incurring only a marginally higher temporal overhead compared to Direct. This implies a commendable balance between effectiveness and temporal efficiency, suggesting a high level of practicality in real-life scenarios.

6.5 Qualitative Analysis with Additional Examples

To provide a more comprehensive understanding of CorpusBrain++, we include a qualitative analysis with additional examples. In this case, a new document titled *Nelson Mandela* arrives in session 1, and the specific text is depicted in Table 7. We provide two real input examples to further explain the mechanism of CorpusBrain++. (i) In the first example, when exposed to an entity-linking query, CorpusBrain++ cannot generate the correct docid *Nelson Mandela* as the top-1 candidate provenance until session 1. Thanks to the high consistency between constructed pseudo-pairs and golden pairs, CorpusBrain++ can perform the CDL task well from session 1. Thanks to the document rehearsal strategy used by CorpusBrain++, it is able to remember the mapping from pseudo-queries of entity linking to the docid *Nelson Mandela* through all sessions. Notably, CorpusBrain++ can retrieve documents related to *Nelson Mandela* such as *Mandla Mandela* (the grandson of *Nelson Mandela*) in session 0, which demonstrates the robustness and generalization capabilities of the CorpusBrain++ framework. (ii) In the second example, when exposed to a QA query, CorpusBrain++ can constantly generate the docid *President of South Africa* related to the question since session 0. Although the document titled *President of South Africa* is not explicitly labeled as the golden provenance, we can find that its content proves beneficial to answer the given question. After session 1, CorpusBrain++ learns the new document titled *Nelson Mandela*, and retrieves *Nelson Mandela* as a top-2 candidate provenance. As both cases illustrate, CorpusBrain++ is capable of retrieving accurate related documents for given downstream queries and maintaining the retrieval capability without catastrophic forgetting.

7 Related Work

In this section, we review three lines of related work, KILTs, GR, and CL.

7.1 KILTs

KILTs refer to a series of language tasks that require extensive and external knowledge sources such as Wikipedia. For instance, fact checking requires the identification of reliable pieces of evidence to establish the authenticity of a claim [56], and open-domain QA entails the need for supporting information from the knowledge base in order to provide an accurate response [16, 24, 29, 62].

Table 7. Case Study Pertaining to a Newly Arrived Document Titled *Nelson Mandela* in Session 1, Herein, We Present Two Real Input Queries in the Test Set and the Corresponding Retrieval Documents of CorpusBrain++ in Different Sessions

Wikipedia Title: Nelson Mandela
Text: Nelson Mandela Nelson Rolihlahla Mandela (18 July 1918–5 December 2013) was a South African anti-apartheid revolutionary, political leader, and philanthropist who served as President of South Africa from 1994 to 1999. He was the country's first black head of state and the first elected in a fully representative democratic election [...]
Input Query 1: (*Entity linking*) [...] Viljoen broke with other right-wing whites in 1994 by taking part in the country's first all-race elections in April of that year, saying the only way to attain self-determination was by cooperating with President `[START_ENT]` Nelson Mandela `[END_ENT]`'s majority African National Congress [...] *Golden Provenance 1*: Nelson Mandela
Retrieval Document 1: Session 0: Nelson Mandela 70th Birthday Tribute [SEP] Mandla Mandela Session 1: Nelson Mandela [SEP] Mandla Mandela Session 2: Nelson Mandela [SEP] Mandla Mandela Session 3: Nelson Mandela [SEP] Mandla Mandela Session 4: Nelson Mandela [SEP] Mandla Mandela
Input Query 2: (*Open-domain QA*) Who succeeded Nelson Mandela as South African president? *Golden Provenance 2*: Nelson Mandela
Retrieval Document 2: Session 0: President of South Africa [SEP] Mandela: Long Walk to Freedom Session 1: President of South Africa [SEP] Nelson Mandela Session 2: President of South Africa [SEP] Nelson Mandela Session 3: President of South Africa [SEP] Nelson Mandela Session 4: President of South Africa [SEP] Nelson Mandela

To facilitate the evaluation of KILTs, a comprehensive benchmark dataset named KILT has been proposed [41], which collects 11 datasets spanning 5 tasks including fact checking, dialogue, slot filling, QA, and entity linking. Essentially, all these tasks in KILT are grounded in the same snapshot of Wikipedia.

Practical solutions to these tasks usually involve a two-step, pipelined framework [5, 16, 24, 29, 62], including a retriever and a reader. Given an input query, a retriever is used to select a limited subset of relevant information from a large knowledge source [5, 31, 41, 46]. Subsequently, a reader is applied to produce the final results by incorporating the input queries and derived support information [30, 31, 42]. The majority of existing approaches in the retrieval component can be divided into two categories: (i) sparse retrieval methods that typically involve constructing an inverted index based on term-based features, and (ii) dense retrieval methods that generally construct a vectorized index based on semantic features and rely on approximate nearest neighbor search algorithms to facilitate efficient retrieval. Very recently, GR methods have been proposed and employed to tackle the retrieval task for KILT [8, 9, 13]. CorpusBrain [9] is an example of this approach; it achieves state-of-the-art retrieval performance.

The majority of prior research on KILTs, including CorpusBrain, is concentrated exclusively on static knowledge source corpus. Nevertheless, in real-world scenarios knowledge accumulates over time, leading to an evolution of the knowledge source corpus. Unfortunately, this pervasive scenario of a dynamic knowledge source corpus has mostly been neglected so far. To the best of our knowledge, our work is the first attempt to focus on the dynamic retrieval scenario for KILTs.

7.2 GR

Traditional methods for IR typically involve a multi-step pipeline paradigm, i.e., the "index-retrieve-then-rank" paradigm [12, 17, 25]. Specifically, the paradigm typically boils down to three sequential steps: (i) creating an index of documents based on their content, (ii) querying the index to retrieve relevant documents, and (iii) ranking the retrieved documents based on their relevance to the query. The pipeline paradigm has stood the test of time due to its adaptability and reliability across applications. Though well-established, the pipeline paradigm encounters several challenges: (i) During training, heterogeneous ranking components are usually difficult to be optimized in an end-to-end way towards the global objective. (ii) During inference, an additional challenge pertains to the substantial memory resource overhead necessary for constructing and maintaining the index, which is a common dilemma not only in the inverted index of sparse retrieval models such as TF-IDF [43] and BM25 [46], but also in the vectorized index of dense retrieval models like DPR [25]. Besides, any errors or inaccuracies introduced during a particular stage can propagate through the system and potentially impact the outcomes of subsequent stages.

To address these disadvantages, GR has been proposed as an alternative paradigm. GR refers to a new retrieval paradigm where a single consolidated model is employed to replace the commonly-used multi-stage search pipeline. With GR, the traditional indexing stage is substituted by indexing documents into model parameters in the model training phase, and the retrieval and ranking stages are replaced by retrieving relevant documents for queries in the model inference phase [36]. In contrast to the classic "index-retrieve-then-rank" paradigm, GR methods exhibit considerable advantages: (i) GR methods allow for end-to-end optimization, hence reducing error propagation. (ii) The lack of constructing large-scale document indexes in GR reduces both time and space overhead. Given the advantages, a surge of explorations of GR methods has recently emerged [4, 13, 55, 67]. GR methods mainly focus on two core issues: (i) how to represent documents with docids, and (ii) how to model the correlation between queries and relevant docids. As for representing docids, three primary techniques are proposed, namely unstructured atomic identifiers (e.g., unique integers [55]), simple string identifiers (e.g., titles [9, 13]), and semantically structured identifiers (e.g., clustering-based representation [55]). Very recently, Wang et al. [59] have proposed neural optimized vocabularial docids, which are learnable by training on the retrieval tasks. Sun et al. [54] have devised a document tokenization learning method to address the challenge of defining document identifiers for GR. As for establishing the semantic mapping from documents to docids, Tay et al. [55] apply memorization-based pre-training to establish a mapping between the content of documents and corresponding docids and retrieval-focused fine-tuning to facilitate the mapping of queries to relevant docids. Chen et al. [9] carefully designs three pre-training tasks to generate pseudo-queries and thus resemble the relevance between downstream queries and docids.

Recently, Mehta et al. [35] have identified the challenge of catastrophic forgetting in DSI while continually indexing new documents, and have proposed DSI++, which incorporates two solutions to alleviate explicit and implicit forgetting, i.e., sharpness-aware minimization and generative memory.

In this work, we have investigated the issue of catastrophic forgetting in the context of CorpusBrain, and have devised solutions to address the problem. Different from DSI++, we (i) focus on continually pre-training rather than simply supervised fine-tuning in DSI++, and (ii) concentrate

on the distinctive scenario of incremental retrieval for KILTs, where queries demonstrate a wider spectrum of diversity in contrast to the traditional retrieval scenario, incorporating varying perspectives such as task, granularity, and complexity. Very recently, Chen et al. [6] and Yoon et al. [64] have also explored how to perform CL for GR over dynamic corpora. Unlike their work, which mainly focuses on a single type of queries, we concentrate on incremental retrieval scenarios for KILTs spanning multiple downstream tasks.

7.3 CL

CL, also commonly referred to as lifelong learning or incremental learning, is a significant and challenging research area that draws inspiration from human cognition, which tends to acquire knowledge in a sequential manner [14]. In contrast to human beings, artificial neural networks often exhibit catastrophic forgetting when confronted with new information, leading to a loss of previously acquired knowledge [28]. Therefore, the CL research area seeks to address this issue by exploring methods to learn from a continuous stream of data while incrementally extending existing knowledge and using it for future learning. The traditional CL scenarios [57] can be grouped into three categories: (i) Task-incremental learning, where models are invariably equipped with task identities conveying the specific task to perform, which are distinct in different sessions. (ii) Domain-incremental learning, where the input distribution keeps changing, while the task structure remains constant in spite of unavailable task identities at test time.

Class-incremental learning, where, without task identities being provided, models must be able to both solve every task seen so far and extrapolate to the tasks encountered. Distinctly, our work concentrates on the CDL task for KILT, where the knowledge source corpus evolves over time without related KILT queries.

To avoid catastrophic forgetting of neural networks, existing methods for CL can be broadly categorized into three families: (i) Replay methods [44, 50] typically involve explicitly retraining the model on a limited subset of samples stored within a memory buffer to alleviate the issue of catastrophic forgetting. To construct the memory buffer, existing work [3, 40, 44, 63] tends to select old training samples or use generative models to produce pseudo-samples. A few principles are employed for sample selection, e.g., iCaRL [44] preserves a representative subset of exemplars per class to effectively approximate the complete data distribution of each class. As for pseudo rehearsal, which is broadly employed in the absence of old samples, the generative models required can be of various categories, e.g., generative adversarial networks [3, 40] and variational autoencoder [63]. (ii) Regularization-based methods [28, 32] incorporate an additional regularization term into the loss function, which restricts the magnitude of representation change during learning on new data, hence consolidating previously acquired knowledge and alleviating catastrophic forgetting. It is common practice in regularization-based methods to maintain a static copy of the previous model for reference purposes. A typical implementation strategy involves the incorporation of a quadratic penalty into the loss function, which imposes a penalty on the variation of parameters that are measured to be important in the previous learning phase [28, 45, 49]. (iii) Parameter isolation methods are usually used in task-incremental learning and typically dedicate an isolated parameter subspace to each task with a suitably designed model architecture. A representative implementation of this model architecture is to explicitly decompose a model into task-sharing and task-specific components. This implementation facilitates explicit modeling of each specific task and hence mitigates catastrophic forgetting of the ability to perform each task. In the context of CDL task for KILT, we take inspiration from this paradigm and propose to use task-sharing components as a form of long-term memory to maintain retrieval capability and task-specific components as a form of short-term memory for the assimilation of newly arriving knowledge.

In this work, we use parameter isolation methods as the primary approach to address the CDL task for KILTs, alongside replay methods to achieve optimal CL effectiveness.

8 Conclusion and Future Work

In this article, we have focused on the dynamic retrieval scenario for KILTs. By defining the CDL task for KILTs and introducing the new benchmark dataset KILT++, our work allows for a systematic and comprehensive assessment of the dynamic retrieval scenario for KILTs.

In particular, we have presented a continual generative pre-training framework for KILTs to address the CDL task. Our framework, CorpusBrain++, allows for effective and efficient CL for KILTs, which results from a synergy between the backbone-adapter architecture and the task-specific pre-training objective tailored for each downstream KILT task. Besides, the framework also incorporates a document rehearsal strategy based on semantic similarity to defy catastrophic forgetting of old documents. Furthermore, a series of extensive experiments validate the effectiveness and efficiency of CorpusBrain++.

Broader Impact. In practical scenarios, user queries often exhibit a broad spectrum of diversity, encompassing various perspectives such as task orientation, granularity, and complexity. To the best of our knowledge, we are the first to explore GR methods in the context of dynamic retrieval scenarios incorporating multiple and diverse downstream tasks. Given that our approach accurately models real retrieval scenarios and offers effectiveness and efficiency advantages, it is well-suited for application in real-world search engines tailored to knowledge-intensive linguistic tasks. We aim for our initial exploration to serve as a benchmark for dynamic retrieval scenarios for KILTs and to inspire the IR community to further enhance the retrieval effectiveness and efficiency in such scenarios.

Limitations and Future Work. Regarding the limitations of our work, we acknowledge that the construction of KILT++ overlooks the temporal order of documents. Consequently, the evaluation might be insufficient when genuinely novel topics and domains enter the corpus. In the future, we will explore new evaluation settings focusing on the OOD issue of topics and domains. Moreover, we currently only consider the continual paradigm of parameter isolation and experience replay. The investigation of alternative continual paradigms such as regularization-based methods, and the exploration of additional categories of parameter isolation methods, are both avenues that merit thorough examination. Despite the promising results of our method, the dynamic retrieval scenario for KILTs presents several unexplored facets, particularly in the era dominated by large language models. One intriguing avenue for exploration involves the design of task-specific pre-training objectives in collaboration with large language models.

Reproducibility. To facilitate reproducibility of the results in this article, we have only used open datasets. The code and constructed benchmark data used to produce our results are available at https://github.com/Sherlock-coder/CorpusBrainPlusPlus.

Acknowledgements

We want to thank our editor and reviewers for their helpful and constructive feedback.

References

[1] Rodrigo B. Almeida, Barzan Mozafari, and Junghoo Cho. 2007. On the evolution of Wikipedia. In *Proceedings of the International AAAI Conference on Web and Social Media (ICWSM)*.

[2] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. 2017. Guided open vocabulary image captioning with constrained beam search. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 936–945.

[3] Ali Ayub and Alan R. Wagner. 2021. EEC: Learning to encode and regenerate images for continual learning. arXiv:2101.04904. Retrieved from https://arxiv.org/abs/2101.04904

[4] Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Scott Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive search engines: Generating substrings as document identifiers. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 35, 31668–31683.

[5] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1870–1879.

[6] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Wei Chen, Yixing Fan, and Xueqi Cheng. 2023. Continual learning for generative retrieval over dynamic corpora. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 306–315.

[7] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yixing Fan, and Xueqi Cheng. 2021. FedMatch: Federated learning over heterogeneous question answering data. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, 181–190.

[8] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yixing Fan, and Xueqi Cheng. 2022. GERE: Generative evidence retrieval for fact verification. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2184–2189.

[9] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yiqun Liu, Yixing Fan, and Xueqi Cheng. 2022. CorpusBrain: Pre-train a generative retrieval model for knowledge-intensive language tasks. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, 191–200.

[10] Herbert H. Clark, S. Haviland, and Roy O. Freedle. 1977. Discourse production and comprehension. In *Discourse Processes: Advances in Research and Theory*. Roy O. Freedle (Ed.), Ablex Publishing Corporation.

[11] Herbert H. Clark and Susan E. Haviland. 1974. Psychological processes as linguistic explanation. In *Explaining Linguistic Phenomena*, 91–124.

[12] Zhuyun Dai and Jamie Callan. 2020. Context-aware term weighting for first stage passage retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1533–1536.

[13] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive entity retrieval. arXiv:2010.00904. Retrieved from https://arxiv.org/abs/2010.00904

[14] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 7 (2021), 3366–3385.

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

[16] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3558–3567.

[17] Jibril Frej, Philippe Mulhem, Didier Schwab, and Jean-Pierre Chevallet. 2020. Learning term discrimination. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1993–1996.

[18] Michael Glass, Alfio Gliozzo, Rishav Chakravarti, Anthony Ferritto, Lin Pan, G. P. Shrivatsa Bhargav, Dinesh Garg, and Avirup Sil. 2020. Span selection pre-training for question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2773–2782.

[19] David Graus, Daan Odijk, and Maarten de Rijke. 2018. The birth of collective memories: Analyzing emerging entities in text streams. *Journal of the Association for Information Science and Technology* 69, 6 (June 2018), 773–786.

[20] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W. Bruce Croft, and Xueqi Cheng. 2020. A deep look into neural ranking models for information retrieval. *Information Processing & Management* 57, 6 (2020), 102067.

[21] Greg Hamerly and Charles Elkan. 2003. Learning the k in k-means. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 16.

[22] Susan E. Haviland and Herbert H. Clark. 1974. What's new? Acquiring new information as a process in comprehension. *Journal of Verbal Learning and Verbal Behavior* 13, 5 (1974), 512–521.

[23] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the International Conference on Machine Learning*. PMLR, 2790–2799.

[24] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1601–1611.

[25] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. arXiv:2004.04906. Retrieved from https://arxiv.org/abs/2004.04906

[26] Pei Ke, Haozhe Ji, Siyang Liu, Xiaoyan Zhu, and Minlie Huang. 2020. SentiLARE: Sentiment-aware language representation learning with linguistic knowledge. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6975–6988.

[27] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv:1412.6980. Retrieved from https://arxiv.org/abs/1412.6980

[28] 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* 114, 13 (2017), 3521–3526.

[29] 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 (2019), 452–466.

[30] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 7871–7880.

[31] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 33, 9459–9474.

[32] Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 12 (2017), 2935–2947.

[33] David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 30.

[34] Arun Mallya and Svetlana Lazebnik. 2018. PackNet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7765–7773.

[35] Sanket Vaibhav Mehta, Jai Gupta, Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Jinfeng Rao, Marc Najork, Emma Strubell, and Donald Metzler. 2022. DSI++: Updating transformer memory with new documents. arXiv:2212.09744. Retrieved from https://arxiv.org/abs/2212.09744

[36] Donald Metzler, Yi Tay, Dara Bahri, and Marc Najork. 2021. Rethinking search: Making domain experts out of dilettantes. *ACM SIGIR Forum* 55, 1 (2021), 1–27.

[37] In Jae Myung. 2003. Tutorial on maximum likelihood estimation. *Journal of Mathematical Psychology* 47, 1 (2003), 90–100.

[38] Rodrigo Nogueira and Jimmy Lin. 2019. From doc2query to docTTTTTquery. In *An MS MARCO Passage Retrieval Task Publication*. University of Waterloo.

[39] Felipe Ortega. 2009. *Wikipedia: A Quantitative Analysis*. Ph.D. Dissertation. Universidad Rey Juan Carlos, Madrid, Spain.

[40] Oleksiy Ostapenko, Mihai Puscas, Tassilo Klein, Patrick Jahnichen, and Moin Nabi. 2019. Learning to remember: A synaptic plasticity driven framework for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11321–11329.

[41] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, et al. 2020. KILT: A benchmark for knowledge intensive language tasks. arXiv:2009.02252. Retrieved from https://arxiv.org/abs/2009.02252

[42] 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 (2020), 1–67.

[43] Juan Ramos. 2003. Using TF-IDF to determine word relevance in document queries. In *Proceedings of the 1st Instructional Conference on Machine Learning*, Vol. 242, 29–48.

[44] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. 2017. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2001–2010.

[45] Hippolyt Ritter, Aleksandar Botev, and David Barber. 2018. Online structured Laplace approximations for overcoming catastrophic forgetting. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 31.

[46] Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval* 3, 4 (2009), 333–389.

[47] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline M. Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Text Retrieval Conference*. Retrieved from https://api.semanticscholar.org/CorpusID:3946054

[48] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. arXiv:1606.04671. Retrieved from https://arxiv.org/abs/1606.04671

[49] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. 2018. Progress & compress: A scalable framework for continual learning. In *Proceedings of the International Conference on Machine Learning*. PMLR, 4528–4537.

[50] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. 2017. Continual learning with deep generative replay. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 30.

[51] Amit Singhal. 2001. Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin* 24, 4 (2001), 35–43.

[52] Asa Cooper Stickland and Iain Murray. 2019. BERT and Pals: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the International Conference on Machine Learning*. PMLR, 5986–5995.

[53] Lixin Su, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Yanyan Lan, and Xueqi Cheng. 2020. Continual Domain Adaptation for Machine Reading Comprehension. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, 1395–1404.

[54] Weiwei Sun, Lingyong Yan, Zheng Chen, Shuaiqiang Wang, Haichao Zhu, Pengjie Ren, Zhumin Chen, Dawei Yin, Maarten de Rijke, and Zhaochun Ren. 2023. Learning to tokenize for generative retrieval. arXiv:2304.04171. Retrieved from https://arxiv.org/abs/2304.04171

[55] Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. 2022. Transformer memory as a differentiable search index. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 35, 21831–21843.

[56] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: A Large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 809–819.

[57] Gido M. Van de Ven and Andreas S. Tolias. 2019. Three scenarios for continual learning. arXiv:1904.07734. Retrieved from https://arxiv.org/abs/1904.07734

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 30 (2017).

[59] Zihan Wang, Yujia Zhou, Yiteng Tu, and Zhicheng Dou. 2023. NOVO: Learnable and interpretable document identifiers for model-based IR. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, 2656–2665.

[60] Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6397–6407.

[61] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. arXiv:2007.00808. Retrieved from https://arxiv.org/abs/2007.00808

[62] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2369–2380.

[63] Fei Ye and Adrian G. Bors. 2022. Task-free continual learning via online discrepancy distance learning. In *Proceedings of the Advances in Neural Information Processing Systems*, Vol. 35 (2022), 23675–23688.

[64] Soyoung Yoon, Chaeeun Kim, Hyunji Lee, Joel Jang, and Minjoon Seo. 2023. Continually updating generative retrieval on dynamic corpora. arXiv:2305.18952. Retrieved from https://arxiv.org/abs/2305.18952

[65] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the International Conference on Machine Learning*. PMLR, 11328–11339.

[66] Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. 2022. Dense text retrieval based on pretrained language models: A survey. arXiv:2211.14876. Retrieved from https://arxiv.org/abs/2211.14876

[67] Yujia Zhou, Zhicheng Dou, and Ji-Rong Wen. 2023. Enhancing generative retrieval with reinforcement learning from relevance feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*.

Received 4 March 2024; revised 16 March 2025; accepted 29 July 2025