Continual Learning for Generative Retrieval over Dynamic Corpora

Jiangui Chen
Ruqing Zhang*
CAS Key Lab of Network Data Science and Technology, ICT, CAS
University of Chinese Academy of Sciences
Beijing, China
{chenjiangui18z,zhangruqing}@ict.ac.cn

Jiafeng Guo†
CAS Key Lab of Network Data Science and Technology, ICT, CAS
University of Chinese Academy of Sciences
Beijing, China
guojiafeng@ict.ac.cn

Maarten de Rijke
University of Amsterdam
Amsterdam, The Netherlands
m.derijke@uva.nl

Wei Chen
CAS Key Lab of Network Data Science and Technology, ICT, CAS
University of Chinese Academy of Sciences
Beijing, China
chenwei2022@ict.ac.cn

Yixing Fan
CAS Key Lab of Network Data Science and Technology, ICT, CAS
University of Chinese Academy of Sciences
Beijing, China
fanyixing@ict.ac.cn

Xueqi Cheng
CAS Key Lab of Network Data Science and Technology, ICT, CAS
University of Chinese Academy of Sciences
Beijing, China
cxq@ict.ac.cn

ABSTRACT

Generative retrieval (GR) directly predicts the identifiers of relevant documents (i.e., docids) based on a parametric model. It has achieved solid performance on many ad-hoc retrieval tasks. So far, these tasks have assumed a static document collection. In many practical scenarios, however, document collections are dynamic, where new documents are continuously added to the corpus. The ability to incrementally index new documents while preserving the ability to answer queries with both previously and newly indexed relevant documents is vital to applying GR models. In this paper, we address this practical continual learning problem for GR. We put forward a novel Continual-LEarner for generatiVE Retrieval (CLEVER) model and make two major contributions to continual learning for GR: (i) To encode new documents into docids with low computational cost, we present Incremental Product Quantization, which updates a partial quantization codebook according to two adaptive thresholds; and (ii) To memorize new documents for querying without forgetting previous knowledge, we propose a memory-augmented learning mechanism, to form meaningful connections between old and new documents. Empirical results demonstrate the effectiveness and efficiency of the proposed model.

CCS CONCEPTS

• **Information systems** → **Retrieval models and ranking**.

KEYWORDS

Document increment, Generative retrieval, Product quantization

ACM Reference Format:
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 (CIKM '23), October 21–25, 2023, Birmingham, United Kingdom.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3583780.3614821

1 INTRODUCTION

Generative retrieval (GR) has emerged as a new paradigm for information retrieval (IR) [34]. Without loss of generality, the GR paradigm aims to integrate all necessary relevant information in the collection into a single, consolidated model. With GR, indexing is replaced by model training, while retrieval is replaced by model inference. A sequence-to-sequence (seq2seq) model is jointly trained for both indexing and retrieval tasks: the indexing task aims to associate the document content with its identifiers (i.e., docids); the retrieval task requires that queries are mapped to relevant docids.

GR and dynamic corpora. Most existing work on GR assumes a stationary learning scenario, i.e., the document collection is fixed [6, 12, 46, 47]. However, dynamic corpora are a common setting for IR. In most real-world scenarios, information changes and new documents emerge incrementally over time. For example, in digital libraries, new electronic collections are continuously added to the system [48]. And a medical search engine may continuously expand its coverage to provide information about emerging diseases, as we have seen with COVID-19 [24]. An important difference between static and dynamic scenarios is that in the former scenario a GR system may be provided with abundant labels for training, but in the latter scenario very few labeled query-document pairs are typically available. Therefore, it is critical to study the continual learning ability of GR models before their use in real-world environments.

*Research conducted when the author was at the University of Amsterdam.
†Jiafeng Guo is the corresponding author.

CIKM '23, October 21–25, 2023, Birmingham, United Kingdom

ACM ISBN 979-8-4007-0124-5/23/10.
https://doi.org/10.1145/3583780.3614821

The *continual document learning* task comes with interesting challenges. For traditional pipeline frameworks for IR [16, 21, 41], indexing and retrieval are two separate modules. Therefore, when new documents arrive, their encoded representations can be directly included in an external index without updating the retrieval model due to the decoupled architecture. In GR, all document information is encoded into the model parameters. To add new documents to the internal index (i.e., model parameters), the GR model must be re-trained from scratch every time the underlying corpus is updated. Clearly, due to the high computational costs, this is not a feasible way of handling a dynamically evolving document collection.

A document-incremental retriever. Our aim is to develop an effective and efficient *Continual-LEarner for generatiVE Retrieval* (CLEVER), that is able to incrementally index new documents while supporting the ability to query both newly encountered documents and previously learned documents. To this end, we need to resolve two key challenges in terms of the indexing and retrieval task.

First, *how to incrementally index new documents with low computational and memory costs?* We introduce *incremental product quantization* (IPQ) based on product quantization (PQ) methods [20] to generate PQ codes for documents as docids, which can represent large volumes of documents via a small number of quantization centroids. The key idea is to incrementally update a subset of centroids instead of all centroids, without the need to update the indices of existing data. Specifically, given the base documents (that is, the initial collection of documents), we iteratively train the document encoder and quantization centroids with a clustering loss and a contrastive loss. The clustering loss offers incentives for representations of documents around a centroid to be close, while the contrastive loss enhances the document representation close to its own random spans. This helps learn discriminative documents and centroid representations, so as to easily generalize to new documents. Then, as new documents arrive, we introduce two adaptive thresholds based on the distances between new and old documents in the representation space, to automatically realize three types of update for centroid representations, i.e., unchanging, changing, and addition. Finally, we index each new document by learning a mapping from document content to its docid.

Second, *how to prevent catastrophic forgetting for previously indexed documents and maintain the retrieval ability?* We take inspiration from the given-new strategy in cognitive science, in which humans attach new information to already known, i.e., given, similar information, in their memory to enhance a mental model of the information as a whole [9, 10, 17]. We propose a memory-augmented learning mechanism to strengthen connections between new and old documents. We first allocate a dynamic memory bank for each session to preserve exemplar documents similar to new documents to prevent forgetting of previously indexed documents. Then, we train a query generator model to sample pseudo-queries for documents and supplement them while continually indexing new documents to prevent forgetting for the retrieval task.

Experimental findings. We introduce two novel benchmark datasets constructed from the existing MS MARCO [36] and Natural Questions [25] datasets, simulating the continual addition of documents to the system. Extensive evaluation shows that CLEVER performs significantly better than prevailing continual learning methods and effectively mitigates catastrophic forgetting in incremental scenarios, while outperforming traditional IR models and existing GR models in non-incremental scenarios.

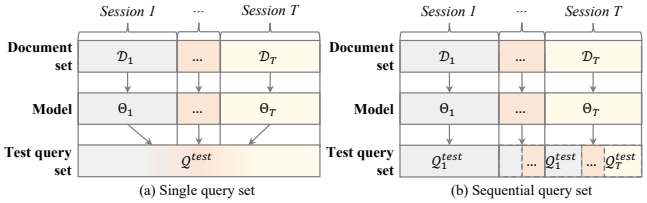


Figure 1: Evaluation criteria.

2 PROBLEM STATEMENT

Task formulation. Given a large-scale base document set $\mathcal{D}_0$ and sufficiently many labeled query-document pairs $\mathcal{P}_0^{\mathcal{D}_0}$, we can train an initial GR model $f(\cdot)$ via a standard seq2seq objective [42]. Let the meta-parameters of the initial model be Θ_0. The continual document learning task assumes the existence of T new datasets $\{\mathcal{D}_1, \ldots, \mathcal{D}_t, \ldots, \mathcal{D}_T\}$, from T sessions arriving in a sequential manner. In any session $t \geq 1$, $\mathcal{D}_t$ is only composed of newly encountered documents $\{d_t^1, d_t^2, \ldots\}$ without queries related to these documents. Let the model parameters before the t-th update be Θ_{t-1}. For session t, the GR model is trained to update its parameters to Θ_t via the new dataset $\mathcal{D}_t$ and previous datasets $\{\mathcal{D}_0, \ldots, \mathcal{D}_{t-1}\}$, and Θ_t serves as input for the datasets $\{\mathcal{D}_0, \ldots, \mathcal{D}_t\}$.

Evaluation. After updating GR models with new documents, we explore two types of test query set for performance evaluation.

Single query set. As illustrated in Figure 1 (a), under this condition, there is only one test query set $\mathcal{Q}^{test}$, and their relevant documents arrive in different sessions. However, we cannot directly compare the retrieval performance before and after incremental updates. The reason is that many widely-used ranking metrics [32] are based on ground-truth relevant documents, which change across sessions. Instead, we compare the overall performance VERT_t of different methods on $\mathcal{Q}^{test}$ in the same session t vertically,

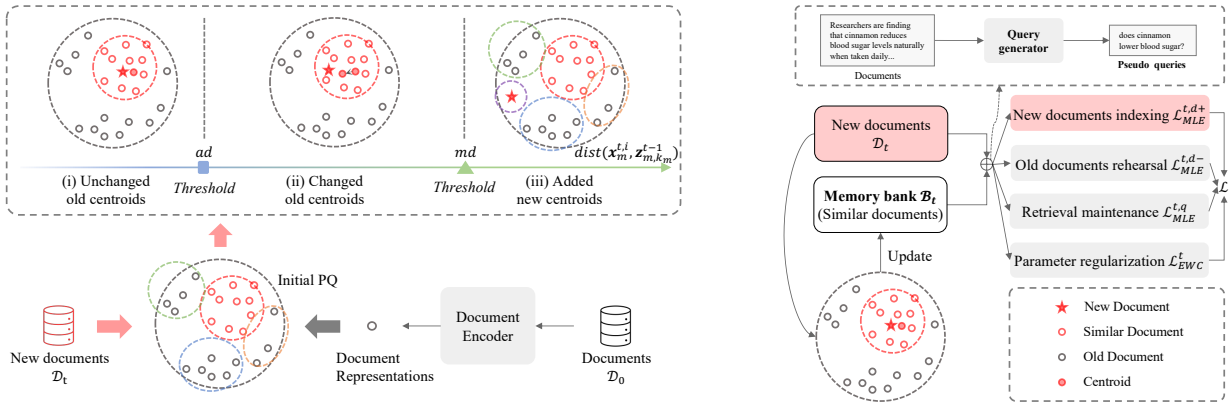
$$\text{VERT}_t = \bigodot_{q \in \mathcal{Q}^{test}, d_q^+ \in \{\mathcal{D}_0, \ldots, \mathcal{D}_t\}} g(d_q^+, f(q; \Theta_t)), \tag{1}$$

where d_q^+ is a relevant document to the query $q \in \mathcal{Q}^{test}$ in existing sessions $\{0, \ldots, t\}$, and $g(\cdot)$ denotes a widely-used evaluation metric for IR; see Section 4.3.

Sequential query set. As illustrated in Figure 1 (b), under this condition, the test query set $\mathcal{Q}_t^{test}$ is specific for each session t, and the relevant documents appear in existing sessions $\{0, \ldots, t\}$. We can directly compare different models across different sessions. Besides VERT, following [26, 33], we apply (i) average performance (AP) to measure the average performance by the end of training with the entire existing data sequence, (ii) backward transfer (BWT) to measure the influence of learning a new session on the preceding sessions' performance, and (iii) forward transfer (FWT) to measure the ability to learn when presented with a new session.

3 METHODOLOGY

In this section, we introduce our *Continual-LEarner for generatiVE Retrieval* (CLEVER). Given an already constructed GR model, we first index newly arrived documents (Section 3.1), and then prevent forgetting of the retrieval ability during incremental indexing (Section 3.2). Figure 2 provides an overview of the method.



(a) Incremental product quantization (IPQ) (b) Overall training objective

Figure 2: (a) Encoding new documents into docids by updating a subset of quantization centroids. (b) Overall training objective for continual indexing while alleviating forgetting of the retrieval ability.

3.1 Indexing new documents

To incrementally index new documents, we need to encode new documents into docids with low computational cost, while learning associations between new documents and their docids.

3.1.1 Incremental product quantization. One popular docid representation is to leverage the *product quantization* (PQ) technique [20] to generate the PQ code as the docid. PQ is able to produce a number of centroids with low storage costs, contributing to representing large collections of documents. However, it is not designed for dynamic corpora. Therefore, we propose *incremental product quantization* (IPQ) based on PQ to represent docids.

The key idea is to design two adaptive thresholds to update a subset of centroids instead of all centroids, without changing the index of the updated centroids. IPQ contains two dependent steps: (i) construct the document encoder and base quantization centroids, from the base documents $\mathcal{D}_0$, and (ii) partially update quantization centroids, based on the relationship between new documents $\mathcal{D}_t$ and old documents $\{\mathcal{D}_0, \ldots, \mathcal{D}_{t-1}\}$.

Building base quantization centroids. Given the base document set $\mathcal{D}_0$, we first leverage BERT [14] as the initial document encoder. Specifically, a special token w_0 = `[CLS]` is added in front of the i-th document $d_0^i = \{w_1, \ldots, w_{|d_0^i|}\}$ in $\mathcal{D}_0$, and the encoder represents the document d_0^i as a series of hidden vectors, i.e., $\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{|d_0^i|}$ = Encoder$(w_0, w_1, \ldots, w_{|d_0^i|})$. We feed the `[CLS]` representation $\mathbf{h}_0$ into a projector network [7, 8], which is a feed-forward neural network with a non-linear activation function (i.e., tanh), to obtain the complete document representation $\mathbf{x}^{0,i}$ of d_0^i.

To better generalize to new documents, we propose a two-step iterative process to iteratively learn document encoder and quantization centroids, to enhance their discriminative abilities. In Step 1, centroids are obtained via a clustering process over document representations, and in Step 2, document representations are learned from centroids with a bootstrapped training process.

Step 1: Clustering process for centroids. Critically, given the D-dimensional document representation $\mathbf{x}^{0,i} \in \mathbb{R}^D$ of the i-th document $d_0^i \in \mathcal{D}_0$, there are three main stages to build centroid representations (more generally the PQ codes) following [20]:

Group division. We use M sub-quantizers to divide the D-dimensional space into M groups, i.e., $\mathbf{x}^{0,i} \in \mathbb{R}^D$ is represented as a concatenation of M sub-vectors $[\mathbf{x}_1^{0,i}, \ldots, \mathbf{x}_m^{0,i}, \ldots, \mathbf{x}_M^{0,i}]$, where $\mathbf{x}_m^{0,i} \in \mathbb{R}^{D/M}$. In this way, the sub-vectors $\{\mathbf{x}_m^{0,i}\}$ of each document $d_0^i \in \mathcal{D}_0$, where $i \in \{1, \ldots, |\mathcal{D}_0|\}$, form the m-th group.

Group clustering. For each group, we apply K-means clustering over all document representations in $\mathcal{D}_0$ to obtain the initial codebook $Z^0 = \{\mathbf{z}_{m,k}^0\}$, where $m \in \{1, \ldots, M\}$, $k \in \{1, \ldots, K\}$, and $\mathbf{z}_{m,k}^0$ is the centroid of the k-th cluster from the m-th group at initial session. The codebook is composed of M sub-codebooks, each of which contains K cluster centroids quantized from a distinct sub-quantizer.

Quantization. Given the sub-vector $\mathbf{x}_m^{0,i} \in \mathbb{R}^{D/M}$, we quantize it to the nearest centroid $\mathbf{z}_{m,k_m}^0$. We select the centroid $\mathbf{z}_{m,\phi(\mathbf{x}_m^{0,i})}^0$ which achieves the minimum quantization error and decide the k_m-th cluster in the m-th group that $\mathbf{x}_m^{0,i}$ belongs to $k_m = \phi(\mathbf{x}_m^{0,i}) = \arg\min_k \|\mathbf{z}_{m,k}^0 - \mathbf{x}_m^{0,i}\|^2$. Finally, document representation $\mathbf{x}^{0,i}$ is quantized as the concatenation of M centroid representations, i.e., $\mathbf{x}^{0,i} = [\mathbf{z}_{1,k_1}^0, \ldots, \mathbf{z}_{m,k_m}^0, \ldots, \mathbf{z}_{M,k_M}^0]$. Accordingly, docid $u_{d_0^i}$ of document d_0^i is obtained by its PQ code $[k_1, \ldots, k_m, \ldots, k_M]$.

Step 2: Bootstrapped training for document representations. High-quality document representation is the foundation of PQ to support effective clustering. However, based on the original BERT, the discriminative ability of these representations may be limited since the representation will focus more on the common words and thus is not differential with other representations [27, 49]. Therefore, we propose a bootstrapped training process based on BERT to learn discriminative document representations. The key idea is to utilize both the contrastive loss and the clustering loss for re-training the BERT encoder itself.

The **contrastive loss** helps to generate the document representation close to its own random spans while being far away from others [27]. We first sample a set of spans at four levels of granularity for each document with length n in $\mathcal{D}_0$, including word-level, phrase-level, sentence-level and paragraph-level: (i) *length sampling*: We first sample the span length from a beta distribution for each level of granularity, i.e., $\ell_{span} = p_{span} \cdot (\ell_{max} - \ell_{min}) + \ell_{min}$, where ℓ_{min} and ℓ_{max} denote the minimum and maximum span length of each level of granularity and p_{span} is sampled by $p_{span} \sim \text{Beta}(\alpha, \beta)$, where α and β are two hyperparameters, and (ii) *position sampling*:

We randomly sample the starting position $start \sim U\ 1, n - \ell_{span}$ and the ending position $end = start + \ell_{span}$. In this way, the final span is denoted as $span = [w_{start}, \ldots, w_{end-1}]$. Given a mini-batch of N documents, we can obtain N whole document representations and their span representations and the contrastive loss is, $\mathcal{L}_{CL} = \sum_{i=1}^{|\mathcal{D}_0|} -\frac{1}{4G} \sum_{s \in S(i)} \log \frac{\exp(\mathrm{sim}(\mathbf{s}_i, \mathbf{s}_s)/\tau)}{\sum_{j=1}^{N*(4G+1)} \mathbb{1}_{[i<j]} \exp(\mathrm{sim}(\mathbf{s}_i, \mathbf{s}_j)/\tau)}$, where G is the number of spans sampled per granularity, $S(i)$ is the index set of spans from d_0^i with size $4G$, $\mathrm{sim}(\cdot)$ is the dot-product function and τ is the temperature hyper-parameter, and $\mathbf{s}_i$ is the span representation, which is computed via the average pooling operation over the output word representations given by the encoder, i.e., $\mathbf{s}_i = \mathrm{AvgPooling}(\mathbf{h}_{start}, \ldots, \mathbf{h}_{end-1})$.

The **clustering loss** computes the mean square error (MSE) between the document representations before and after quantization, enabling to cluster document representations around the centroid representations. Concretely, the MSE loss $\mathcal{L}_{MSE}$ is, $\mathcal{L}_{MSE} = \sum_{i=1}^{|\mathcal{D}_0|} \|\mathbf{x}^{0,i} - \hat{\mathbf{x}}^{0,i}\|^2$. We then re-train the BERT encoder via $\mathcal{L}_{CL} + \mathcal{L}_{MSE}$, and also adopt the `[CLS]` representation given by the re-trained encoder as the document representation.

Repeating Step 1 and Step 2. Step 1 and Step 2 are repeated iteratively for v epochs. Finally, we obtain the initial quantization centroids to build the PQ codes $\mathcal{U}_0^{\mathcal{D}_0} = \{u_{d_0^1}, \ldots, u_{d_0^{|\mathcal{D}_0|}}\}$ for $\mathcal{D}_0$, and the Encoder$(\cdot)$ learned on $\mathcal{D}_0$ is fixed for later sessions.

Adaptively updating quantization centroids. With the arrival of new documents $\mathcal{D}_t$ during session t, we first utilize the learned Encoder$(\cdot)$ to obtain the document representations. Based on the representations of new and old documents, a simple method is to re-cluster them to obtain the novel PQ codes as the docids. However, this may incur high computational cost in updating all clustering results and re-training the GR model based on the updated docids.

Ideally, we have a way to balance the trade-off between update efficiency and quantization error. Here, we introduce a partial codebook update strategy for this purpose. Specifically, we design three types of update for centroid representations in each sub-codebook, contributing to efficiency in memory and computational load:

- **Unchanged old centroids**: It is pointless to update the centroid when the features in some groups of new documents have a trivial contribution in their corresponding centroid update.
- **Changed old centroids**: It is possible that some features of new documents have a vital contribution to a centroid update.
- **Added new centroids**: We should add new centroids when new documents are significantly different from all old documents.

To achieve the above update, we first divide the representation vector $\mathbf{x}^{t,i}$ of each document $d_t^i \in \mathcal{D}_t$, into M sets of sub-vectors and add each sub-vector $\mathbf{x}_m^{t,i}$ to the corresponding m-th group. Then, for each sub-codebook, we compute the Euclidean distance [11] between the newly arrived sub-vector $\mathbf{x}_m^{t,i}$ and its nearest centroid $\mathbf{z}_{m,k_m}^{t-1}$ based on the last session $t-1$, i.e., $\mathrm{dist}(\mathbf{x}_m^{t,i}, \mathbf{z}_{m,k_m}^{t-1})$. Finally, we devise two adaptive thresholds, i.e., ad and md, according to this distance, to achieve three types of update.

For each cluster in a sub-codebook, ad is the average distance between each document sub-vector and the quantization centroid,

$$ad = \frac{1}{|C_{m,k}^{t-1}|} \sum_{j=1}^{|C_{m,k}^{t-1}|} \mathrm{dist}(\mathbf{x}_m^{t,j}, \mathbf{z}_{m,k_m}^{t-1}), \tag{2}$$

where $C_{m,k}^{t-1}$ is the set of document indices assigned to the centroid $\mathbf{z}_{m,k}^{t-1}$. And md is the maximum distance between each document sub-vector and the quantization centroid, denoted as,

$$md = \max_{j \in C_{m,k}^{t-1}} \mathrm{dist}(\mathbf{x}_m^{t,j}, \mathbf{z}_{m,k_m}^{t-1}) + \mathrm{rand\_dist}, \tag{3}$$

where $\mathrm{rand\_dist} \sim U(0, ad)$ is sampled from the continuous uniform distribution. Note that the condition $ad \leq md$ always holds.

Therefore, as depicted in Figure 2(a), we can automatically decide the update type of each centroid representation as follows:

- If $\mathrm{dist}(\mathbf{x}_m^{t,i}, \mathbf{z}_{m,k_m}^{t-1}) \ddot{Y}\ ad$, the centroid remains unchanged.
- Alternatively, if $ad \leq \mathrm{dist}(\mathbf{x}_m^{t,i}, \mathbf{z}_{m,k_m}^{t-1}) \leq md$, we need to update the centroid representation. We first update the set via $C_{m,k}^t = C_{m,k}^{t-1} \cup \{ind\}$, where ind is the index number of $\mathbf{x}_m^{t,i}$. Then each centroid can be updated by, $\mathbf{z}_{m,k_m}^t = \mathbf{z}_{m,k_m}^{t-1} + \frac{1}{|C_{m,k}^t|}(\mathbf{x}_m^{t,i} - \mathbf{z}_{m,k_m}^{t-1})$.
- Finally, if $\mathrm{dist}(\mathbf{x}_m^{t,i}, \mathbf{z}_{m,k_m}^{t-1})\ \ddot{\imath}\ md$, we add a new cluster and thus there are $K+1$ clusters in the group. We directly use the document sub-vector as the centroid representation, i.e., $\mathbf{z}_{m,K+1}^t = \mathbf{x}_m^{t,i}$.

After applying the above update strategy for all M sub-codebooks, we obtain the specific codebook Z^t at session t: (i) for new documents in $\mathcal{D}_t$, we obtain their PQ codes $\mathcal{U}_t^{\mathcal{D}_t}$ based on the Z^t as the docids, and (ii) for old documents, their PQ codes will not be affected since we only operate on the centroid representations, instead of the index of the updated centroid. In the case of old documents around a centroid sharing the same representation, i.e., $ad = md = 0$, we directly change the centroid representation based on the new document sub-vector.

3.1.2 Indexing objective. To memorize information about each new document, we leverage maximum likelihood estimation (MLE) [35] to maximize the likelihood of a docid conditioned on the corresponding document, i.e.,

$$\mathcal{L}_{MLE}^{t,d+} = \sum_{\mathcal{D}_t, \mathcal{U}_t^{\mathcal{D}_t}} \log p(u_{d_t^i}, \Theta_t \mid d_t^i; \Theta_{t-1}), \tag{4}$$

where $d_t^i \in \mathcal{D}_t$, $u_{d_t^i} \in \mathcal{U}_t^{\mathcal{D}_t}$, Θ_t is the GR model parameters at the session t, and $i \in \{1, \ldots, |\mathcal{D}_t|\}$.

3.2 Preserving retrieval ability

During continual indexing of new documents, it is important for the GR models to prevent forgetting the retrieval ability. We are inspired by the fact that humans benefit from previous similar experiences when taking actions [9, 10, 17] and propose a memory-augmented learning mechanism to build meaningful connections between new and old documents. Specifically, we first construct a memory bank with similar documents for each new session and replay the process of indexing them alongside the indexing of new documents. Then, we leverage a query generator model to sample pseudo-queries for documents and the resulting query-docid pairs are employed to maintain the retrieval ability. The overall learning process is visualized in Figure 2 (b).

Dynamic memory bank construction. The memory bank is allocated to store a tiny subset of old documents which are similar to new documents in the PQ space. We assume that two documents are similar if many dimensions of their PQ codes are the same. For each document in $\mathcal{D}_t$, we target to retrieve its similar documents at different levels. Concretely, we iteratively change its PQ code at different dimensions, which includes the following steps: (i) we first

set the number o of PQ code dimensions that will be changed to 1; (ii) we randomly select o dimensions of the PQ code and assign different centroids to the selected dimensions to obtain the similar PQ code. We repeat this process c times; and (iii) we obtain the similar documents from the previous sessions if they are associated with the obtained PQ codes. The processes in (ii) and (iii) are repeated by increasing o with 1 to at most $M/6$.

Finally, we group the similar documents of each document in $\mathcal{D}_t$ to construct a specific memory bank $\mathcal{B}_t$ at the session t. Note that the memory bank is dynamically updated at each new session.

Rehearsing the indexing of old documents. For each new session t, we aim to prevent forgetting previously indexed documents. Given the meta model parameters Θ_{t-1} before the t-th update, we apply MLE over the memory bank $\mathcal{B}_t$ to update the GR model, i.e.,

$$\mathcal{L}^{t,d-}_{MLE} = \sum_{d^i_t \in \mathcal{B}_t, u_{d^i_t} \in \mathcal{U}^{\mathcal{B}_t}_t} \log p(u_{d^i_t}, \Theta_t \mid d^i_t; \Theta_{t-1}), \quad (5)$$

where $\mathcal{U}^{\mathcal{B}_t}_t$ is the PQ codes of $\mathcal{B}_t$, and $i \in \{1, \ldots, |\mathcal{B}_t|\}$.

Constructing pseudo query-docid pairs. To prevent forgetting the retrieval ability during indexing new documents, we train a query generator model to sample pseudo-queries for documents and supplement the query-docid pairs during indexing. We fine-tune the T5 model [38] based on the query-document pairs $\mathcal{P}^{\mathcal{D}_0}_0$ in the initial session, by taking the document terms as input and producing a query following [37]. After fine-tuning, the model parameters of the query generator Θ_{qg} are fixed.

For each new session t, we generate pseudo-queries for each document in $\mathcal{D}_t$ and $\mathcal{B}_t$ via Θ_{qg} and denote the obtained pairs of pseudo-queries and documents as $\mathcal{P}^{\mathcal{D}_t}_t$ and $\mathcal{P}^{\mathcal{B}_t}_t$, respectively. Given the meta model parameters Θ_{t-1} of the GR model, we also apply MLE to maximize the likelihood of a relevant docid conditioned on each pseudo query in $\mathcal{P}^{\mathcal{D}_t}_t$ and $\mathcal{P}^{\mathcal{B}_t}_t$, denoted as,

$$\mathcal{L}^{t,q}_{MLE} = \sum_{\{\mathcal{P}^{\mathcal{D}_t}_t, \mathcal{P}^{\mathcal{B}_t}_t\}, \{\mathcal{U}^{\mathcal{D}_t}_t, \mathcal{U}^{\mathcal{B}_t}_t\}} \log p(u_{d^{\psi(q_j)}_t}, \Theta_t \mid q_j; \Theta_{t-1}), \quad (6)$$

where $(q_j, d^{\psi(q_j)}_t) \in \{\mathcal{P}^{\mathcal{D}_t}_t, \mathcal{P}^{\mathcal{B}_t}_t\}$, $\psi(q_j)$ is the index of the relevant document to q_j and $j \in \{1, \ldots, |\{\mathcal{P}^{\mathcal{D}_t}_t, \mathcal{P}^{\mathcal{B}_t}_t\}|\}$. $u_{d^{\psi(q_j)}_t} \in \{\mathcal{U}^{\mathcal{D}_t}_t, \mathcal{U}^{\mathcal{B}_t}_t\}$ is the relevant docid.

3.3 Overall training objective

In the training phase, we sequentially train the GR model on each session t by combining the objective for indexing and retrieval, i.e.,

$$\min_{\Theta_t} -(\mathcal{L}^{t,d+}_{MLE} + \mathcal{L}^{t,d-}_{MLE} + \mathcal{L}^{t,q}_{MLE}) + \lambda \mathcal{L}^t_{EWC}, \quad (7)$$

where λ is a hyper-parameter. The elastic weight consolidation (EWC) [23] loss $\mathcal{L}^t_{EWC}$ is used to regularize the model parameters, via the weighted distance between Θ_{t-1} and Θ_t,

$$\mathcal{L}^t_{EWC} = \sum_l F_l \left(\Theta_{t-1,l} - \Theta_{t,l}\right)^2, \quad (8)$$

where F is the Fisher information matrix [23], and F_l denotes each model parameter.

4 EXPERIMENTAL SETTINGS

Next, we summarize our experimental settings. The code can be found at https://github.com/ict-bigdatalab/CLEVER.

4.1 Benchmark construction

To facilitate the study of continual document learning for GR, we build two benchmark datasets, i.e., CDI-MS and CDI-NQ, from MS MARCO Document Ranking [36] and Natural Questions (NQ) [25], respectively. MS MARCO contains 367,013 query-document training pairs, 3,213,835 documents, and 5,192 queries in the dev set. NQ contains 307k query-document training pairs, 231k documents, and 7.8k queries in the dev set. We report the performance results on the dev sets as both MS MARCO Document Ranking and NQ leaderboard limit the frequency of submissions [33, 46].

To mimic the new arrival of documents in MS MARCO and NQ, we first randomly sample 60% documents from the whole document set as the base documents $\mathcal{D}_0$, and leverage their corresponding relevance labels to construct the query-document pairs $\mathcal{P}^{\mathcal{D}_0}_0$. Then, we randomly sample 10% documents from the remaining document set as the new document set, and this operation is repeated for 4 times to obtain $\mathcal{D}_1$–$\mathcal{D}_4$. The test query set is defined as follows: (i) for a single query set, all dev queries are denoted as $\mathcal{Q}^{test}$, and (ii) for sequential query set, we sample 60%, 10%, 10%, 10% and 10% queries from the whole dev query set as $\mathcal{Q}^{test}_0, \ldots, \mathcal{Q}^{test}_4$, respectively.

4.2 Models

4.2.1 Baselines. **Traditional IR models.** (i) **BM25** [41] is an effective sparse retrieval model and we re-index all previously seen documents upon the arrival of new documents. (ii) **DPR** [21] is a representative dense retrieval model with BERT-based dual-encoder architecture. We use the model trained on the first session $\mathcal{D}_0$ to encode newly arrived documents and then add their encoded representations to the existing external index.

Generative retrieval models. (i) **DSI** [46] encodes all information about the corpus in the model parameters and we adopt atomic docids in DSI. (ii) **DSI-QG** [56] leverages a query generation model to augment the document collection at indexing. (iii) **NCI** [47] utilizes a prefix-aware weight-adaptive decoder and we adopt NCI with DocT5Query for augmented queries. (iv) **Ultron** [54] applies a three-stage training pipeline and we adopt Ultron with PQ as the docid. Due to their limitations in accommodating dynamic corpora, we only evaluate the performance in non-incremental scenarios.

Furthermore, we compare our model with an adaption of Ultron as the **BASE** method, wherein PQ technique is used to generate docids and the GR model is continually fine-tuned by directly mapping each new document to its docid. We also compare with **DSI++** [33], which continuously fine-tunes DSI over new documents by directly assigning each new document an atomic docid, i.e., an arbitrary unique integer. We re-implement it since the source code has not yet been released.

4.2.2 Model variants. To verify the effectiveness of IPQ, we implement variants with the memory-augmented learning mechanism. To build base quantization centroids, we have (i) **CLEVER**$_{atomic}$, which uses arbitrary unique integers as docids, as used in DSI++; (ii) **CLEVER**$_{PQ}$, which directly uses the original BERT$_{base}$ [14] to obtain document representations, and builds PQ codes as docids by the original PQ technique [20]; the codebook is fixed in all sessions; and (iii) **CLEVER**$_{PQ+Re}$, which extends CLEVER$_{PQ}$ by re-clustering the document representations obtained by BERT$_{base}$ as new documents arrive; the codebook is updated at each new session.

To adaptively update the quantization centroids, the variants are: (i) **CLEVER**$_{PQ+Dis}$ leverages the two-step iterative process to build discriminative base PQ codes; then, the codebook is fixed for new sessions, i.e., only adopting the "unchanged old centroids" type; (ii) **CLEVER**$_{PQ+Dis+ad}$ extends **CLEVER**$_{PQ+Dis}$ by adding the *ad* threshold, i.e., adopting "unchanged old centroids" and "changed old centroids;" (iii) **CLEVER**$_{PQ+Dis+md}$ extends **CLEVER**$_{PQ+Dis}$ by adding the *md* threshold, i.e., adopting "added new centroids" and "changed old centroids" to update the quantization centroids; and (iv) **CLEVER**$_{PQ+Dis+Re}$ extends **CLEVER**$_{PQ+Dis}$ by re-building discriminative PQ codes for all documents as new documents arrive; the codebook is updated at each new session.

To verify the effectiveness of the memory-augmented learning mechanism, variants (while using IPQ) are: (i) **CLEVER**$^{-EWC}$, which removes $\mathcal{L}^t_{EWC}$ in Eq. 7 to re-train the GR model; (ii) **CLEVER**$^{-MLE(d-)}$, which removes $\mathcal{L}^{t,d-}_{MLE}$ in Eq. 7 to re-train the GR model; (iii) **CLEVER**$^{-MLE(q)}$, which removes $\mathcal{L}^{t,q}_{MLE}$ in Eq. 7 to re-train the GR model; and (iv) **CLEVER**^{Random}, which randomly selects some old documents to construct the memory bank with the same number of similar documents in CLEVER, which is an adaption of DSI++ [33], and then re-trains the GR model via Eq. 7.

4.3 Evaluation metrics

The evaluation metric $g(\cdot)$ for IR in Section 2 is usually taken to be mean reciprocal rank (MRR@N), recall (R@N), hit ratio (Hits@N) and top-N retrieval accuracy (ACC@N). Following [33, 46, 47, 54], we show the continual results in terms of MRR@10 and HIT@10 for CDI-MS and CDI-NQ, respectively. By conducting further analyses, we find that the relative order of different models on other IR metrics is quite consistent with that on the MRR@10 and Hits@10.

4.4 Implementation details

For IPQ, the length M of PQ codes is 24, the number of clusters K is 256, and the dimension of vectors D is 768. For the contrastive loss in the document encoder, ℓ_{min} and ℓ_{max} for sampling phrase-level spans are 4 and 16, respectively. For sentence-level spans, ℓ_{min} and ℓ_{max} are 16 and 64, respectively. For paragraph-level spans, ℓ_{min} and ℓ_{max} are 64 and 128, respectively. The α and β in the beta distribution are 4 and 2, respectively, which skews sampling towards longer spans. The number of spans sampled per granularity G is 5. For the memory-augmented learning mechanism, the repeat time c is 10, the probability p is 0.2, the scale r is 0.2, and λ is 0.5.

To train the document encoder in IPQ, we initialize the document encoder from the official BERT's checkpoint. We use a learning rate of $5e^{-5}$ and Adam optimizer [22] with a linear warmup over the first 10% steps. Long input documents are truncated into several chunks with a maximum length of 512. The hyper-parameter of τ is 0.1. We train for 6 epochs on four NVIDIA Tesla A100 40GB GPUs.

The GR baselines and all variants of CLEVER, are based on the transformer-based encoder-decoder architecture, where the hidden size is 768, the feed-forward layer size is 3072, the number of transformer layers is 12, and the number of self-attention heads is 12, for both encoder and decoder. We implement the generative model in PyTorch based on Huggingface's Transformers library. We initialize the parameters of the encoder-decoder architecture from the official checkpoint of T5$_{base}$ [38]. We use a learning rate of $3e^{-5}$ and Adam optimizer with the warmup technique, where the learning rate increases over the first 10% of batches, and then decays linearly to zero. The max length of the input is 512, the label smoothing is 0.1, the weight decay is 0.01, and the gradient norm clipping is 0.1. We train in batches of 8192 tokens on four NVIDIA Tesla A100 40GB GPUs. At inference time, we adopt constrained beam search [12] to decode the docids with 24 timesteps and 15 beams. To train the query generator, we also initialize the parameters from the official checkpoint of T5$_{base}$ [38], with a learning rate of $5e^{-4}$. For each new document, we adopt beam search to decode the pseudo queries with utmost 32 timesteps and 10 beams.

5 EXPERIMENTAL RESULTS

In this section, we (i) analyze the retrieval performance on the CID-MS and CID-NQ datasets under both *incremental* and *non-incremental* settings, (ii) assess catastrophic forgetting and forward transfer abilities, and (iii) analyze the memory and computation cost. For (ii) and (iii), we conduct experiments on the CID-MS dataset under *sequential query set* setting in terms of VERT(%).

5.1 Baseline comparison

Incremental performance on a single query set. The performance comparisons between CLEVER and baselines on the *single query set* are shown in Table 1. BM25 exhibits better performance than DPR and the underlying reason may be that BM25 is a data-independent probabilistic model, which renders it adaptable in the face of dynamic corpora. And the BASE method suffers a significant drop as new documents are added. By assigning new documents with atomic docids and using sampled old documents, DSI++ shows slight improvements over BASE. These results show that continual document learning for GR is a non-trivial challenge.

When we look at variants of CLEVER in terms of IPQ, we find that: (i) CLEVER$_{PQ}$ performs worse than CLEVER$_{atomic}$ which updates the embeddings for each individual docid (also used in DSI++), showing that it is difficult for the GR models to accommodate to new documents without updating docids. However, as shown in Section 5.4, for CLEVER$_{atomic}$, its memory continues to grow with the increase of documents and the time consumption of each update step increases with the number of steps. (ii) CLEVER$_{PQ+Re}$ and CLEVER$_{PQ+Dis+Re}$ give the worst performance. The reason might be that re-clustering old and new documents changes the previously learned centroids and thus the docids of old documents, which makes the learned document-docid mapping invalid. (iii) The improvements of CLEVER$_{PQ+Dis}$ over CLEVER$_{PQ}$ demonstrate the need to learn discriminative document representations and quantization centroids. (iv) CLEVER$_{PQ+Dis+ad}$ and CLEVER$_{PQ+Dis+md}$ outperform CLEVER$_{PQ+Dis}$, showing that incorporating updates to old centroids and introducing new centroids could facilitate the assimilation of new documents.

When we look at variants of CLEVER with different learning mechanisms, we observe that: (i) CLEVER$^{-MLE(q)}$ performs worse than CLEVER$^{-MLE(d-)}$ and CLEVER^{-EWC}, showing that constructing pairs of pseudo queries and docids and supplementing them during continual indexing contributes to preventing forgetting for the retrieval ability. (ii) CLEVER^{Random} shows the worst performance. Randomly selected old documents do not provide

Table 1: Model performance under the *single query set* setting. We evaluate the performance of Q^{test} in each session $\mathcal{D}_0$–$\mathcal{D}_4$ in terms of VERT (%), respectively. $*$ indicates statistically significant improvements over all baselines (p-value $\ddot{Y}$ 0.05).

	CDI-MS (MRR@10)					CDI-NQ (Hits@10)				
Model	$\mathcal{D}_0$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	$\mathcal{D}_0$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$
BM25	28.47	29.12	29.33	29.24	28.96	35.65	35.46	35.16	36.88	35.23
DPR	38.51	34.25	28.07	26.34	23.68	40.37	36.92	31.10	28.64	26.78
BASE	42.78	39.51	38.66	36.16	34.93	67.13	64.45	62.08	59.24	58.61
DSI++	42.54	41.60	41.12	39.58	38.29	66.05	65.66	64.47	63.51	63.26
CLEVER$_{atomic}$	42.54	42.52	41.73	41.80	41.21	66.05	66.37	66.58	65.82	65.41
CLEVER$_{PQ}$	42.78	41.53	40.97	39.10	38.04	67.13	66.09	64.39	64.26	63.57
CLEVER$_{PQ+Re}$	42.78	40.39	39.15	37.18	36.48	67.13	65.30	64.15	63.26	62.33
CLEVER$_{PQ+Dis}$	44.96	42.58	41.60	40.26	39.50	68.74	67.02	65.47	64.95	64.22
CLEVER$_{PQ+Dis+ad}$	44.96	44.03	43.78	42.81	42.06	68.74	67.43	67.69	66.30	66.15
CLEVER$_{PQ+Dis+md}$	44.96	43.81	43.59	42.30	41.74	68.74	66.85	66.34	65.92	65.28
CLEVER$_{PQ+Dis+Re}$	44.96	42.27	40.53	38.64	37.90	68.74	65.60	64.28	63.79	62.83
CLEVER^{-EWC}	44.96	43.51	42.67	41.27	41.30	68.74	67.16	66.84	66.75	66.09
CLEVER$^{-MLE(d-)}$	44.96	42.55	42.01	40.87	40.33	68.74	66.81	66.07	65.39	64.83
CLEVER$^{-MLE(q)}$	44.96	42.07	41.51	39.52	39.47	68.74	66.02	66.11	64.94	64.28
CLEVER^{Random}	44.96	41.83	41.24	39.76	38.62	68.74	65.37	64.24	63.81	63.10
CLEVER	**44.96***	**45.36***	**44.81***	**44.07***	**43.75***	**68.74***	**68.25***	**68.36***	**67.71***	**67.50***

Table 2: Model performance under the *sequential query set* setting. We evaluate the performance of Q_0^{test}, …, Q_4^{test} in each session $\mathcal{D}_0$–$\mathcal{D}_4$ in terms of VERT (%), respectively. For AP, BWT and FWT, ↑ indicates higher is better, and ↓ indicates lower is better. $*$ indicates statistically significant improvements over all baselines (p-value $\ddot{Y}$ 0.05).

	CDI-MS (MRR@10)								CDI-NQ (Hits@10)							
Model	$\mathcal{D}_0$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	**AP↑**	**BWT↓**	**FWT↑**	$\mathcal{D}_0$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$	**AP↑**	**BWT↓**	**FWT↑**
BM25	29.63	29.54	28.30	28.12	28.37	27.99	1.00	28.58	34.93	33.67	33.58	32.83	33.64	32.60	1.42	33.43
DPR	40.48	35.61	33.73	30.07	26.75	31.54	2.24	31.54	39.56	37.24	35.93	32.96	30.71	33.03	2.81	34.21
BASE	43.64	40.08	37.91	36.55	35.27	32.91	7.23	37.45	68.43	65.38	63.19	60.57	59.02	56.55	8.46	62.04
DSI++	43.25	41.39	40.58	39.24	38.31	38.31	2.74	39.81	68.21	66.83	64.75	62.30	61.47	62.30	3.02	63.84
CLEVER$_{atomic}$	43.25	42.51	42.49	41.67	40.74	40.05	2.61	41.85	68.21	67.54	67.03	66.84	66.40	65.60	2.00	66.95
CLEVER$_{PQ}$	43.64	42.23	41.37	40.81	39.44	38.29	4.01	40.96	68.43	65.78	64.33	63.50	63.44	61.97	3.91	64.21
CLEVER$_{PQ+Re}$	43.64	41.92	41.03	38.41	37.16	36.95	4.36	39.63	68.43	65.28	63.05	62.83	62.41	61.02	4.22	63.39
CLEVER$_{PQ+Dis}$	45.30	43.26	42.70	41.57	40.21	39.47	3.93	41.93	69.57	66.54	65.84	64.01	64.27	62.23	3.53	65.17
CLEVER$_{PQ+Dis+ad}$	45.30	44.83	44.65	43.27	43.19	43.45	0.99	43.98	69.57	68.48	67.50	67.83	66.99	66.97	1.38	67.70
CLEVER$_{PQ+Dis+md}$	45.30	44.07	43.27	43.35	42.62	42.67	1.32	43.33	69.57	68.02	67.34	66.04	65.85	66.01	1.69	66.81
CLEVER$_{PQ+Dis+Re}$	45.30	42.83	41.04	39.59	38.77	38.26	4.06	40.56	69.57	67.31	65.48	64.02	63.25	63.02	3.64	65.02
CLEVER^{-EWC}	45.30	43.71	43.56	42.98	42.35	42.45	1.41	43.15	69.57	67.01	68.35	67.24	66.90	66.43	1.73	67.38
CLEVER$^{-MLE(d-)}$	45.30	43.40	42.76	42.03	42.11	41.58	1.92	42.58	69.57	68.74	67.52	66.36	65.38	65.10	3.02	67.00
CLEVER$^{-MLE(q)}$	45.30	42.81	42.09	41.74	41.88	40.80	2.45	42.13	69.57	66.32	65.61	65.06	64.40	63.56	3.29	65.35
CLEVER^{Random}	45.30	42.22	41.53	40.71	39.50	39.74	2.69	40.99	69.57	65.01	63.48	62.26	61.79	61.43	3.75	63.14
CLEVER	**45.30***	**45.26***	**45.09***	**44.85***	**44.71***	**44.39***	**0.82***	**44.98***	**69.57***	**69.04***	**69.21***	**69.36***	**68.75***	**68.56***	**0.78***	**69.09***

insights into the new documents and may introduce noise for continual indexing. (iii) CLEVER^{-EWC} performs worse than CLEVER, showing the effectiveness of limiting the scope of model updates.

Finally, CLEVER achieves the best performance. The results imply that applying an adaptive update strategy for PQ codes can assign effective docids to new documents without changing the old docids. And rehearsing old similar documents and generating pseudo queries, can actively absorb knowledge from new documents while preserving previously learned retrieval ability.

Incremental performance on a sequential query set. The performance comparisons between CLEVER and the baselines on the *sequential query set* are shown in Table 2 (recall that the metrics were introduced in Section 2). The relative order of different models under this setting in terms of *VERT* is quite consistent with that on the previous setting of a single query set. For the evaluation metrics that measure the performance across different sessions in terms of AP, BWT and FWT, the full version of CLEVER achieves the best performance, again demonstrating the effectiveness of the proposed IPQ and learning mechanisms.

Table 3: Model performance in non-incremental scenarios. We train on $\mathcal{D}_0$ and evaluate the performance of Q^{test}. Note that DSI++ trained on $\mathcal{D}_0$ is DSI. $*$ indicates statistically significant improvements over all baselines (p-value $\ddot{Y}$ 0.05).

Model	**CDI-MS (MRR@10)**	**CDI-NQ (Hits@10)**
BM25	28.47	35.65
DPR	38.51	40.37
DSI	42.54	66.05
DSI-QG	42.62	66.80
NCI	43.14	67.49
Ultron	42.78	67.13
CLEVER	**44.96***	**68.74***

Non-incremental performance. To assess the performance of CLEVER before getting into the incremental aspect, we evaluate the performance of Q^{test} on $\mathcal{D}_0$ under the single query set setting. As shown in Table 3, we see that: (i) Compared with traditional IR models, CLEVER and existing generative retrieval methods achieve better performance, indicating the effectiveness of integrating different components into a single consolidated model. (ii) CLEVER achieves better results than existing generative retrieval models,

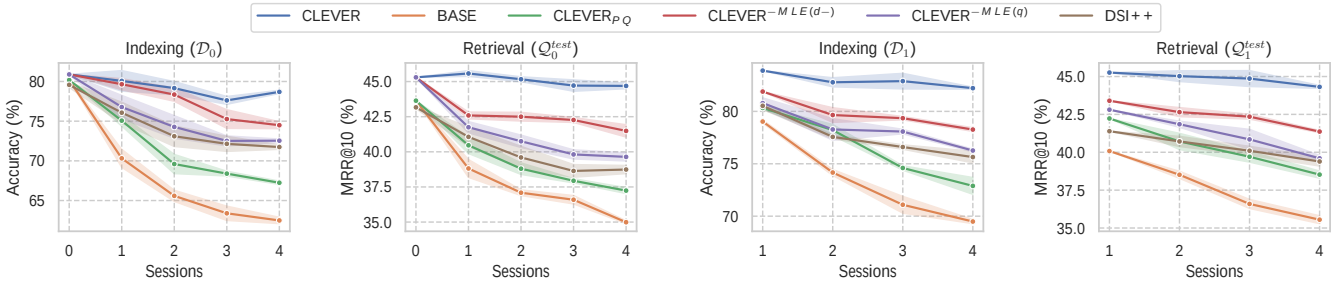


Figure 3: The catastrophic forgetting phenomenon of GR models. Based on the CDI-MS dataset, we illustrate the indexing accuracy of $\mathcal{D}_0$ and $\mathcal{D}_1$, and the retrieval MRR@10 of Q_0^{test} and Q_1^{test} under sequential query set setting.

Table 4: Forward transfer analysis on CDI-MS under sequential query set. We evaluate the performance of Q_0^{test}–Q_4^{test} in terms of VERT (%), respectively. $*$ indicates statistically significant improvements over all baselines (p-value $\ddot{Y}$ 0.05).

Model	$\mathcal{D}_0$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_4$
INDIVIDUAL	40.72	39.38	40.61	38.57	38.04
CLEVER$_{init}$	45.30	40.63	41.15	40.94	40.22
CLEVER	**45.30**	**45.26***	**45.09***	**44.85***	**44.71***

demonstrating that the two-step iterative process to learn discriminative PQ codes as docids contributes to the retrieval effectiveness.

5.2 Assessing catastrophic forgetting

To assess catastrophic forgetting of proposed methods, we show how the performance of the base session $\mathcal{D}_0$ and first incremental learning session $\mathcal{D}_1$ varies over the training process on the remaining sessions. For the indexing task, we evaluate the overall indexing accuracy of $\mathcal{D}_0$ and $\mathcal{D}_1$, i.e., we take a document as input of the GR model, if the generated sequence exactly matches with the correct docid, we treat the document as a positive sample. Otherwise, the document is a negative sample. For the retrieval task, we evaluate the performance of Q_0^{test} and Q_1^{test}, i.e., MRR@10. See Figure 3.

We observe that: (i) CLEVER$_{PQ}$, CLEVER$^{-MLE(d-)}$ as well as CLEVER$^{-MLE(q)}$ suffer from catastrophic forgetting. Applying IPQ and the memory-augmented learning mechanism separately does not provide sufficient assurance for the model to perform well during continual document learning. (ii) DSI++ underperforms CLEVER by a large margin. A possible reason is that the atomic integers used in DSI++ as docids are difficult to quickly adapt to new documents and have a large impact on the docids of old documents, which may result in the loss of previously learned knowledge. (iii) Compared to CLEVER, the phenomenon of catastrophic forgetting is not as well mitigated in CLEVER$^{-MLE(d-)}$ and CLEVER$^{-MLE(q)}$, which underlines the importance of rehearsing old documents and the generated pseudo-queries. And (iv) CLEVER almost avoids catastrophic forgetting on both indexing and retrieval tasks, showing its effectiveness in a dynamic setting.

5.3 Assessing forward transfer

Positive forward knowledge transfer is an essential ability during continual document learning including indexing and retrieval. Therefore, in this section, we explore the forward transfer ability of the CLEVER model, i.e., transferring knowledge from old documents to new documents. For CLEVER without initialization from the previous sessions we write CLEVER$_{init}$ and for individually fine-tuning the GR model on each session we write INDIVIDUAL. Table 4 displays the results. We see that: (i) CLEVER consistently and significantly outperforms INDIVIDUAL in the last four sessions. The underlying reason may be that CLEVER transfers old knowledge to new settings when continuously indexing new documents. INDIVIDUAL learns the indexing and retrieval tasks from each new session independently, which has a small size of data. (ii) The performance improvements of CLEVER over CLEVER$_{init}$ further demonstrate the need for prior initialization in CLEVER, i.e., initializing new model parameters from the last parameters.

5.4 Effectiveness-efficiency trade-off

We evaluate the effectiveness and efficiency of different models on Q_0^{test} in terms of VERT(%) after training all sessions. Regarding training time, we compare the overall training time by the end of the sequential training. For memory footprint, we compute the disk space occupied by the model at the training end. Figure 4 shows the relative memory and training time, i.e., the memory ratio and the time ratio of these methods with respect to BASE, respectively.

Figure 4 (a) shows the effectiveness-memory trade-off. We find that: (i) Traditional IR models (BM25 and DPR) consume much larger memory footprints than generative IR models, without discernible advantages in retrieval performance. This result indicates the significant memory consumption of re-computing representations and re-indexing for new documents. (ii) Although CLEVER$_{atomic}$ and DSI++ are much more effective than the BASE model and CLEVER$_{PQ+Re}$, they suffer from severe memory inefficiency since they need the large softmax output space that comes with atomic docids and the embedding for each individual docid must be added to the model as new documents arrive. (iii) CLEVER performs best in effectiveness and is almost as efficient as the BASE model. CLEVER only occupies a small amount of additional memory compared to BASE, which does not grow over sessions.

Figure 4 (b) shows the effectiveness-training time trade-off. We observe that: (i) BM25 exhibits a swift training process. However, its performance may be deemed suboptimal due to its vulnerability to the vocabulary mismatch issue, as well as its inability to adequately encapsulate semantic information. (ii) The BASE model achieves training acceleration at the cost of compromised performance, which suggests that maintaining effectiveness is a non-trivial challenge for GR in dynamic corpora setting. (iii) CLEVER$_{atomic}$ and DSI++ sacrifice training time for effectiveness since the randomly initial embeddings require training from scratch. (iv) CLEVER$_{PQ+Re}$ re-trains all the centroids every session, leading to some computational overhead. However, the performance is still not improved

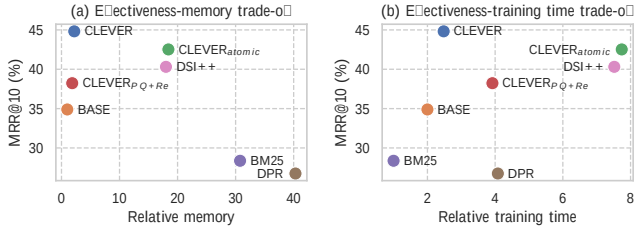


Figure 4: Comparison on (a) effectiveness-memory trade-off and (b) effectiveness-training time trade-off. Up and left is better. The search is performed on GPU.

too much. (v) CLEVER performs best in effectiveness and requires similar training times as the BASE model. These results demonstrate that CLEVER can be well deployed in practical environments due to its high effectiveness and efficiency.

6 RELATED WORK

Generative retrieval. Recently, generative retrieval (GR) has been proposed as an alternative paradigm [34] for IR. Unlike the traditional "index-then-rank" paradigm [14–16, 21, 28, 29, 31, 41, 51], a single seq2seq model is used to directly map a query to a relevant docid. In this way, the large external index is transferred into an internal index (i.e., model parameters), simplifying the data structure for supporting the retrieval process. Besides, it enables the end-to-end optimization towards the global objective for IR tasks.

GR is related to two key issues: (i) the *indexing task*: how to associate the content of each document with its docid, and (ii) the *retrieval task*: how to map the queries to their relevant docids. For the indexing task, previous efforts can boil down to two research lines. One is to generate docids for documents [53] including atomic identifiers (e.g., a unique integer identifier [46]), simple string identifiers (e.g., titles [5, 12, 44], n-grams [2, 4] or URLs [54]) and semantically structured identifiers (e.g., clustering-based prefix representation [46] or PQ [54]). The other is to establish a semantic mapping from documents to docids. Various kinds of document content have been proposed to enhance the association [6, 46, 54], e.g., contexts at different semantic granularities [6, 55] and hyperlink information [6]. For the retrieval task, most approaches directly learn to map queries to relevant docids in an autoregressive way. Recently, some work has been adopted to generate pseudo-queries [47, 54] and designed pre-training tasks [6] to tackle the limited availability of labeled data. However, current GR methods mainly focus on a stationary corpus scenario, i.e., with a fixed document collection.

Very recently, Mehta et al. [33] have shown that continually memorizing new documents leads to considerable forgetting of old documents. They directly assigned each new document an arbitrary unique integer identifier, and randomly sampled some old documents using experience replay [3] for incremental updates. However, learning embeddings for each individual new docid from scratch incurs prohibitively high computational costs, while the relationships between new and old documents may not be easily obtained from randomly-selected exemplars. And they only considered the sequential query set setting for performance evaluation.

In this work, the proposed IPQ technique is able to effectively represent new documents by updating a subset of centroids instead of all centroids, eliminating the need to update existing data indices. In IPQ, the partial codebook update strategy can be applied to other clustering-based docids, e.g., clustering-based prefix representation in DSI [46], which we leave as future work.

Continual learning. Continual learning (CL) has been a long-standing research topic to overcome the catastrophic forgetting problem of previously acquired knowledge, while continuously learning new knowledge from few labeled samples [43]. Recently, CL has been considered in computer vision [40, 45] and natural language processing [1, 19], but few efforts have been devoted to IR so far. CL scenarios [30, 39] can be divided into task increment, domain increment and class increment. In this work, we consider the practical setting of dynamic corpora with newly added documents.

Existing CL approaches [13] can be categorized into: (i) replay methods, maintaining a subset of previous samples and training models together with samples in the new session [18, 52]; (ii) regularization-based methods, regularizing the model parameters to enable important parameters concerning the previous tasks to be protected when training on each new task [23, 50]; and (iii) parameter-isolation methods, dynamically allocating a set of parameters for each task [1]. Here, we take advantage of replay methods and regularization-based methods to memorize new documents.

7 CONCLUSION

In this work, we have focused on a critical requirement for generative retrieval (GR) models to be usable in practical scenarios, where new documents are continuously added to the corpus. In particular, we have presented a continual learning method to alleviate possible high computational costs for generating new docids, and leverage both past similar documents and pseudo-queries for consolidating knowledge. Extensive experiments have demonstrated the effectiveness and efficiency of our method.

Despite the promising results that GR has shown, its scalability remains a challenging issue, particularly concerning document addition, removal, and updates. These factors significantly impact the widespread adoption of GR in various applications. For the proposed CLEVER method, exploring the joint optimization of quantization methods in IPQ and GR models using supervised labels, and devising advanced thresholds for adaptively updating PQ codes, hold great potential for enhancing retrieval effectiveness.

ACKNOWLEDGMENTS

This work was funded by the National Natural Science Foundation of China (NSFC) under Grants No. 62006218 and 61902381, the China Scholarship Council under Grants No. 202104910234, the Youth Innovation Promotion Association CAS under Grants No. 2021100, the project under Grants No. JCKY2022130C039 and 2021QY1701, the CAS Project for Young Scientists in Basic Research under Grant No. YSBR-034, the Innovation Project of ICT CAS under Grants No. E261090, and the Lenovo-CAS Joint Lab Youth Scientist Project. This work was also (partially) funded by the Hybrid Intelligence Center, a 10-year program funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organisation for Scientific Research, https://hybrid-intelligence-centre.nl, and project LESSEN with project number NWA.1389.20.183 of the research program NWA ORC 2020/21, which is (partly) financed by the Dutch Research Council (NWO). All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. 2017. Expert Gate: Lifelong Learning with a Network of Experts. In *CVPR*. 3366–3375.

[2] Michele Bevilacqua, Giuseppe Ottaviano, Patrick Lewis, Wen tau Yih, Sebastian Riedel, and Fabio Petroni. 2022. Autoregressive Search Engines: Generating Substrings as Document Identifiers. In *NeurIPS*.

[3] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. 2019. On Tiny Episodic Memories in Continual Learning. *arXiv preprint arXiv:1902.10486* (2019).

[4] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Maarten de Rijke, Yiqun Liu, Yixing Fan, and Xueqi Cheng. 2023. A Unified Generative Retriever for Knowledge-Intensive Language Tasks via Prompt Learning. In *SIGIR*.

[5] Jiangui Chen, Ruqing Zhang, Jiafeng Guo, Yixing Fan, and Xueqi Cheng. 2022. GERE: Generative Evidence Retrieval for Fact Verification. In *SIGIR*. 2184–2189.

[6] 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 *CIKM*. 191–200.

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*, Vol. 119. PMLR, 1597–1607.

[8] Xinlei Chen and Kaiming He. 2021. Exploring Simple Siamese Representation Learning. *CVPR* (2021), 15745–15753.

[9] Herbert H Clark, S Haviland, and Roy O Freedle. 1977. Discourse Production and Comprehension.

[10] Herbert H Clark and Susan E Haviland. 1974. Psychological Processes as Linguistic Explanation. *Explaining linguistic phenomena* (1974), 91–124.

[11] Per-Erik Danielsson. 1980. Euclidean Distance Mapping. *CGIP* (1980).

[12] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive Entity Retrieval. In *ICLR*.

[13] 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. *PAMI* 44, 7 (2021), 3366–3385.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. 4171–4186.

[15] Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, and Alfio Gliozzo. 2021. Robust Retrieval Augmented Generation for Zero-shot Slot Filling. In *EMNLP 2021*. 1939–1949.

[16] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *CIKM*. 55–64.

[17] Susan E Haviland and Herbert H Clark. 1974. What's New? Acquiring New Information as a Process in Comprehension. *JVLBA* 13, 5 (1974), 512–521.

[18] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the Knowledge in a Neural Network. *stat* 1050 (2015), 9.

[19] 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 *ICML*. 2790–2799.

[20] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product Quantization for Nearest Neighbor Search. *PAMI* 33, 1 (2010), 117–128.

[21] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *EMNLP*. 6769–6781.

[22] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[23] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the National Academy of Sciences* 114, 13 (2017).

[24] Kayvan Kousha and Mike Thelwall. 2020. COVID-19 Publications: Database Coverage, Citations, Readers, Tweets, News, Facebook Walls, Reddit Posts. *Quantitative Science Studies* 1, 3 (2020), 1068–1091.

[25] 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), 453–466.

[26] David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient Episodic Memory for Continual Learning. *NeurIPS* 30 (2017).

[27] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, and Xueqi Cheng. 2022. Pre-train a Discriminative Text Encoder for Dense Retrieval via Contrastive Span Prediction. In *SIGIR*. 848–858.

[28] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Xiang Ji, and Xueqi Cheng. 2021. PROP: Pre-training with Representative Words Prediction for Ad-hoc Retrieval. In *WSDM*. 283–291.

[29] Xinyu Ma, Jiafeng Guo, Ruqing Zhang, Yixing Fan, Yingyan Li, and Xueqi Cheng. 2021. B-PROP: Bootstrapped Pre-training with Representative Words Prediction for Ad-hoc Retrieval. In *SIGIR*. 1513–1522.

[30] Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. 2022. Online Continual Learning in Image Classification: An Empirical Survey. *Neurocomputing* 469 (2022), 28–51.

[31] Jean Maillard, Vladimir Karpukhin, Fabio Petroni, Wen-tau Yih, Barlas Oguz, Veselin Stoyanov, and Gargi Ghosh. 2021. Multi-Task Retrieval for Knowledge-Intensive Tasks. In *ACL*. 1098–1111.

[32] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

[33] 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 preprint arXiv:2212.09744* (2022).

[34] 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.

[35] In Jae Myung. 2003. Tutorial on Maximum Likelihood Estimation. *Journal of Mathematical Psychology* 47, 1 (2003), 90–100.

[36] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated Machine Reading Comprehension Dataset. In *CoCo@NIPs*.

[37] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document Expansion by Query Prediction. *arXiv preprint arXiv:1904.08375* (2019).

[38] 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.

[39] Rahul Ramesh and Pratik Chaudhari. 2021. Model Zoo: A Growing Brain That Learns Continually. In *ICLR*.

[40] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. iCaRL: Incremental Classifier and Representation Learning. In *CVPR*. 2001–2010.

[41] Stephen Robertson and Hugo Zaragoza. 2009. *The Probabilistic Relevance Framework: BM25 and Beyond*. Now Publishers Inc.

[42] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. *NeurIPS* (2014).

[43] Zhen Tan, Kaize Ding, Ruocheng Guo, and Huan Liu. 2022. Graph Few-shot Class-incremental Learning. In *WSDM*. 987–996.

[44] Yubao Tang, Ruqing Zhang, Jiafeng Guo, Jiangui Chen, Zuowei Zhu, Shuaiqiang Wang, Dawei Yin, and Xueqi Cheng. 2023. Semantic-Enhanced Differentiable Search Index Inspired by Learning Strategies. *arXiv preprint arXiv:2305.15115* (2023).

[45] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. 2020. Few-shot Class-incremental Learning. In *CVPR*.

[46] Yi Tay, Vinh Q 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 *NeurIPS*.

[47] Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, et al. 2022. A Neural Corpus Indexer for Document Retrieval. *Advances in Neural Information Processing Systems* 35 (2022), 25600–25614.

[48] Ian H. Witten, David Bainbridge, and David M. Nichols. 2010. *How to Build a Digital Library*. Morgan Kaufmann.

[49] Shicheng Xu, Liang Pang, Huawei Shen, and Xueqi Cheng. 2022. Improving Multi-task Generalization Ability for Neural Text Matching via Prompt Learning. *arXiv preprint arXiv:2204.02725* (2022).

[50] Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual Learning through Synaptic Intelligence. In *ICML*. PMLR, 3987–3995.

[51] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing Dense Retrieval Model Training with Hard Negatives. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1503–1512.

[52] Bowen Zhao, Xi Xiao, Guojun Gan, Bin Zhang, and Shu-Tao Xia. 2020. Maintaining Discrimination and Fairness in Class Incremental Learning. In *CVPR*.

[53] Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. 2022. Dense Text Retrieval Based on Pretrained Language Models: A Survey. *arXiv preprint arXiv:2211.14876* (2022).

[54] Yujia Zhou, Jing Yao, Zhicheng Dou, Ledell Wu, Peitian Zhang, and Ji-Rong Wen. 2022. Ultron: An Ultimate Retriever on Corpus with a Model-based Indexer. *arXiv preprint arXiv:2208.09257* (2022).

[55] Yu-Jia Zhou, Jing Yao, Zhi-Cheng Dou, Ledell Wu, and Ji-Rong Wen. 2023. DynamicRetriever: A Pre-trained Model-based IR System Without an Explicit Index. *Machine Intelligence Research* 20, 2 (2023), 276–288.

[56] Shengyao Zhuang, Houxing Ren, Linjun Shou, Jian Pei, Ming Gong, Guido Zuccon, and Daxin Jiang. 2022. Bridging the Gap between Indexing and Retrieval for Differentiable Search Index with Query Generation. *arXiv preprint arXiv:2206.10128* (2022).