ECAI 2024
U. Endriss et al. (Eds.)

© 2024 The Authors.
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License 4.0 (CC BY-NC 4.0).
doi:10.3233/FAIA240971

BiLEE: Bi-Level Early Exiting for Generative Document Retrieval

Rui Fang[a,*], **Chin-Yuan Yeh**[a], **Hsi-Wen Chen**[a] **and Ming-Syan Chen**[a]

[a]National Taiwan University

Abstract. Generative document retrieval (GDR) uses pre-trained Transformer-based large language models (LLMs) to extract contextual information and directly predict document identifier token sequences, outperforming traditional document retrieval methods. However, LLMs incur significant computational costs, hindering GDR's practical application and making inference acceleration essential. Early exiting is one of the conditional computing techniques that expedites LLM inference, but it faces challenges when integrated into GDR due to GDR's semantically hierarchical structured identifiers, which cause error amplification from premature exits. Moreover, although beam search expands the search space, the hierarchical structure of document identifiers restricts the diversity of initial tokens, leading to inefficiencies. In this work, we introduce *Bi-Level Early Exiting for Generative Document Retrieval (BiLEE)*, comprising *Layer Level Early Exiting (LLEE)* and *Token Level Early Exiting (TLEE)*. LLEE are designed for hierarchical document identifiers, dynamically escaping from the middle layer of the Transformer calculation based on a data-driven calibrated token threshold. TLEE exiting from unpromising candidate sequences, thus discarding unpromising search beams and enhancing beam search efficiency. Both components dynamically balance the speed-to-accuracy trade-offs for different token positions, doubling GDR's inference speed and obtaining $13\times$ reduction for FLOPs while maintaining the same level of accuracy. Source code: https://github.com/Rui-Fang/BiLEE.

1 Introduction

Document retrieval [29, 2, 38, 33] is a technique critical to search engines, question-answering, and dialog systems. Past document retrieval methods [29, 2, 38, 33] can be categorized into sparse retrieval (SR) and dense retrieval (DR). However, both approaches have their limitations. On the one hand, SR [27, 26], based on keyword-matching, assumes word overlaps between queries and relevant documents and often fails to grasp semantic subtleties. On the other hand, DR [20] addresses the limitations of SR by adopting an embedding technique to explore the document semantic rather than lexical matching, which improves the likelihood of finding relevant results. However, when a query is related to multiple documents in the semantic space, DR struggles to retrieve all relevant documents by a single embedding representation [33, 36].

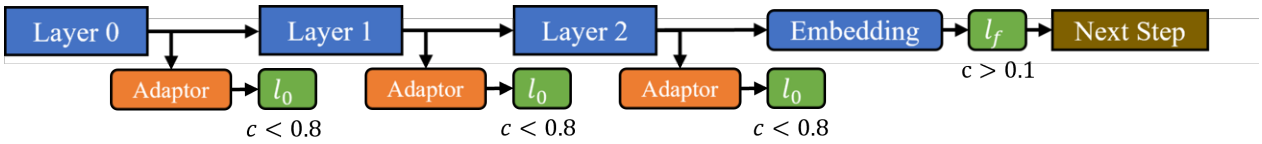
Recently, *generative document retrieval (GDR)*, utilizing pre-trained *large language models* (LLMs) to explore document semantics, has gained significant attention [29, 39, 33]. In GDR, LLMs function as document repositories, with the training process implicitly memorizing and facilitating connections between queries and documents via the attention mechanism in Transformers [31], automatically generating *identifiers* that points to documents relevant to the given queries [22, 10]. Unlike natural language generation tasks, a sequence of tokens for GDR represents a path along a top-down hierarchical structure based on the taxonomy of the documents. Each token from start to end of the sequence represents levels of subclasses from top to bottom of the hierarchical structure. A sequence of tokens is denoted as a *docid* (*document identifiers)*.[1] In the inference stage of a GDR model, Beam search [17] is employed to simultaneously explore multiple sequences of tokens (termed beam hypotheses), that is, partially constructed *docid*s, and selects the top candidates to predict the next token. With beam search, GDR avoids being trapped in a local optimum when predicting *docid*. Note that document retrieval for downstream tasks such as web search or retrieval-then-rank that requires numerous ranked candidate documents usually requires a larger beam search size.

Although GDR demonstrates potential advantages over traditional document retrieval methods, its computational demands are substantial [33], primarily due to the Transformer-based LLM architecture. Moreover, while beam search allows for comprehensive exploration of the *docid* search space, it further increases computational complexity, presenting a clear trade-off between the beam search size and efficiency. To address this challenge, the *Early Exiting* strategy [30, 28] is a promising way to reduce the computational for multi-layered LLMs, where the number of layers required for correct token prediction can dynamically adjust based on the difficulty of each token. The core idea of early exiting is to calculate a *confidence level* for individual layers during the transformer's layer-by-layer computation, determining whether to bypass the subsequent layers (i.e., exit) or to continue for the inference calculation of the current token. If the exit point is correctly chosen, exiting the layer-by-layer computation early allows LLMs to generate the correct token using the output from an intermediate layer rather than the final layer of a transformer, thereby conserving computational resources.

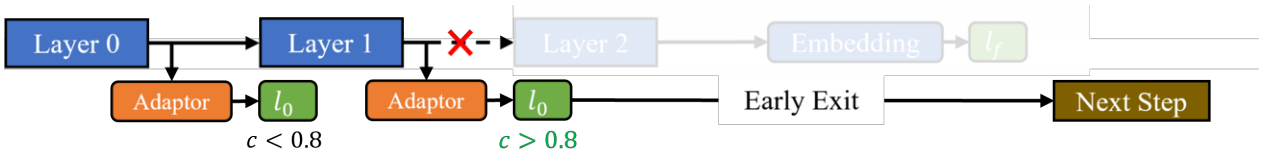
A prior work by Schuster et al. [28] has explored adopting early exiting on LLMs for language generation tasks. However, several challenges arise when applying early exiting to GDR due to the distinct characteristics of *docid*s required for GDR. Firstly, the hierarchical structure of document categorization in a *docid* leads to a cascading effect where errors in earlier tokens significantly im-

* Corresponding Author. Email: rfang@arbor.ee.ntu.edu.tw

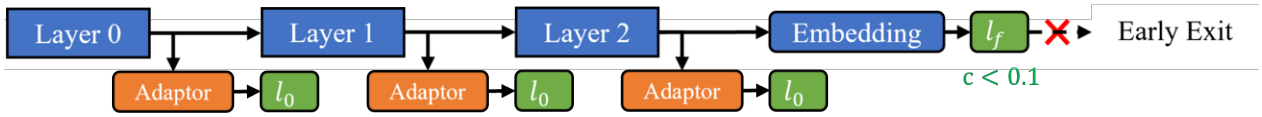
[1] The hierarchical k-means algorithm is commonly used to cluster documents into *docid* based on their semantic correlation. When the prefix digits of *docid* are identical, it indicates a higher relevance between the two documents.



(a) Conventional GDR inference with multi-layered transformer.



(b) Layer Level Early Exiting (LLEE).



(c) Token Level Early Exiting (TLEE).

Figure 1. Illustration of BiLEE during a token generation process for GDR. (a) depicts the case where neither TLEE nor LLEE is triggered; (b) presents LLEE: when an adaptor finds confidence score surpasses the designated layer threshold λ_l, the calculation can exit at this layer; (c) If the overall confidence score for the final embedding is below the token threshold λ_t, the search for this token will be removed (exited) among the beam search candidates.

pact subsequent tokens' accuracy. Intuitively, such a condition requires a higher precision for earlier tokens than for later ones. However, we observe that the prediction difficulty of the end of a *docid* sequence also increases since these tokens require differentiating between smaller semantic differences within finer categories. Such striking characteristics cause token prediction difficulty to fluctuate based on *token position*, deviating from a simple monotonic trend. As a result, since prior work [28] assumes a monotonic decrease in token difficulty with sentence length, it produces sub-optimal outcomes when applied to GDR. In addition, the smaller vocabulary size of GDR causes the conventional approach of fine-tuning the entire model to be prone to overfitting.

Secondly, beam search results also create redundancy because the diversity of initial tokens in these search results is not required for GDR. Since *docid* is hierarchically structured, most beam hypotheses during the initial $1 \sim 2$ tokens explorations exhibit a very low score and will be discarded in the later decoding stages, resulting in a significant waste of compute. Nevertheless, the usual layer-wise confidence levels of these low-score hypotheses are usually too low to meet the layer-level early exit thresholds, necessitating alternative early exiting designs.

To address the aforementioned challenges, we introduce an acceleration framework designed for GDR: *Bi-Level Early Exiting for Generative Document Retrieval (BiLEE)*. As illustrated in Figure 1, BiLEE prevents redundant computations for highly confident tokens at the layer level for accelerating token inference with *Layer Level Early Exiting (LLEE)* and reduces redundant computations for low-confidence tokens at the token level by identifying and terminating less likely *docid* candidates within the beam search process with *Token Level Early Exiting (TLEE)*.

Specifically, through a data-driven calibration process, LLEE establishes the thresholds for each token to better adapt to the varying difficulty of different token positions, facilitating early exiting for each token position without compromising the overall performance. Besides, to tackle the overfitting problem caused by the smaller token vocabulary size, LLEE employs layer-wise adaptor training processes instead of the conventional approach of fine-tuning the entire model (see Figure 1(b) and Figure 2). Simultaneously, TLEE presents an early exiting strategy for the beam search process by preemptively terminating beam hypothesis branches based on beam confidence scores (*i.e.*, relevance scores); we minimize inefficiencies within the beam search space and reinforce the efficacy of our early exit strategies (see Figure 1(c)).

During the inference phase, our model leverages LLEE to exit early from layers exhibiting high confidence and utilizes TLEE to exit from low-confidence branches during beam search. Experimental results demonstrate that BiLEE reduced the FLOPs usage of GDR inference by up to $12.59\times$ and enhanced throughput by up to $2.53\times$, all while maintaining accuracy levels without notable degradation.

Our contributions are summarized as follows:

- We introduce *Bi-Level Early Exiting for Generative Document Retrieval (BiLEE)*, representing the inaugural accelerated GDR approach employing early exiting strategies. To our knowledge, this is the first work to accelerate GDR via early exiting.
- We introduce a bi-level acceleration framework: *Layer Level Early Exiting (LLEE)* operates at the layer level to skip redundant computations for individual token predictions, while *Token Level Early Exiting (TLEE)* focuses on exiting low-confidence tokens and narrowing the beam search space.
- Combining LLEE and TLEE, BiLEE yields up to $4.34\times$ reduction in FLOPs with a $1.92\times$ increase in throughput acceleration on the NQ320k dataset and up to $12.59\times$ reduction in FLOPs with a $2.53\times$ increase in throughput on the TriviaQA dataset.

2 Related Work

2.1 Generative Document Retrieval

Document retrieval facilitates the location of relevant documents within a database or corpus based on user queries, with various applications, including web searches [19], question answering [12], and dialogue systems [3]. Traditional sparse retrieval (SR) methods

[27, 26, 25] rely on word overlaps between queries and documents, often faltering due to lexical mismatches. Dense retrieval (DR) methods [12, 34, 24] employ encoders for semantic matching, mitigating the zero-recall phenomenon. However, a single query representation may struggle to recall all relevant documents, as they may not align closely in the semantic space [36]. Moreover, DR methods may not fully exploit the capabilities of Large Language Models (LLMs) to understand semantics. Recently, Generative Document Retrieval (GDR) has emerged as a paradigm within DR, directly mapping input queries to corresponding document identifiers using generative models. The pioneering work GENRE [5] generates document titles using constrained beam search. Subsequent works employ various strategies for *docid*, such as filenames [5], random numbers [29], single atomic embeddings [22, 21], and multi-view identifiers [15]. While these identifiers either cannot map semantically similar documents together or require a vast vocabulary to map the entire corpus, a recent line of studies employs trained LLM to generate semantically clustered *docid* in a Seq2seq manner. For instance, DSI [29] assigns hierarchically and semantically clustered docids to documents, while DSI-QG [39] enhances DSI by generating pseudo queries using doc2query. NCI [33] and RIPOR [35] also introduce improvements by generating pseudo queries, incorporating location offset prefixes to *docid*, and employing prefix-aware weight-adaptive decoders. However, the practical adoption of GDR faces limitations. LLMs entail significant computational demands, necessitating large model scales to memorize corpus information [22] effectively. Additionally, GDR requires large beam sizes to generate multiple candidate documents, resulting in significant computational requirements.

2.2 Early Exiting for LLM

Early Exiting [30] dynamically adjusts computational resources for tokens, offering simplicity and effectiveness. Initially developed for convolutional neural networks, it terminates computation once the model achieves sufficient confidence, reducing inference resources and time [18]. While extensively applied to encoder-only transformer models, especially in the BERT series [8, 37, 16], generalizing it to Encoder-Decoder poses challenges due to structural and paradigmatic differences. CALM [28] pioneers the application of Early Exiting on Encoder-Decoder Transformers, addressing crucial issues like layer-level early exit strategies and linking sequence-level constraints with individual per-token exit decisions. SkipDecode [6] proposes a uniform exit point at each token position for batch inference in Decoder-only models but lacks adaptability to token complexity variations. Chen et al. [4] introduces a framework named EE-LLM for large-scale training and inference of Early-Exit LLMs, tackling scalability challenges. However, the above methods employ exit thresholds decreasing monotonically with sentence length, failing to address token difficulty diversity in GDR outputs and requiring whole model fine-tuning, leading to overfitting. In contrast, our BiLEE utilizes finer-grained thresholds and uses the adaptors rather than fine-tuning the whole model to resolve these issues. Additionally, we extend Early Exit to the beam search process, addressing search space waste in GDR beam search by calibrating token-level exit thresholds.

3 Preliminaries

To ensure documents with similar semantics have close document identifiers (*docids*), GDR with semantic ID utilizes a hierarchical k-means algorithm. Each document d in the corpus $\mathcal{C}$ is initially encoded by a pre-trained LLM [13] to obtain a vector embedding. Subsequently, the k-means algorithm is recursively applied to these representations, forming a hierarchical clustering [33]. For clusters with fewer than c documents at any level, k-means clustering is reapplied to create k sub-clusters within that cluster. This hierarchical clustering results in a tree structure where each document d resides in a leaf node, establishing a deterministic path $\mathbf{r} = [r_0, r_1, \ldots, r_m]$, with $r_i \in [0, k), \forall, i < m$ and $r_m \in [0, c)$, which serves as the *docid* for each document d.

An encoder-decoder transformer model is trained to autoregressively generate a sequence of *docid* from a given query q. Specifically, each token r_i at position i in the *docid* sequence is iteratively predicted based on the query q and the sequence of previously predicted tokens up to position i, denoted as $\mathbf{r}_{<i}$. This prediction is facilitated by the transformer's output and a position-specific embedding matrix W_i. Formally, the hidden state output h_i^L for the ith token from the final decoder layer L is represented as

$$h_i^L = \text{Decoder}(\text{Encoder}(q), \mathbf{r}_{<i}, \theta), \tag{1}$$

where θ denotes the model parameters. The prediction probability p can be expressed as follows.

$$p(r_i|q, \mathbf{r}_{<i}, \theta) = \text{SoftMax}(W_i h_i^L), \tag{2}$$

where $\mathbf{r}_{<i}$ denotes the predicted *docid* tokens up to position i. With early exiting, prediction probabilities are calculated using decoder layers earlier than L. Specifically, the hidden state from a mid-layer j is denoted as h^j.

Finally, the relevance of a *docid* sequence $\mathbf{r}_n$ up to position n is quantified by a relevance score S_n.

$$S_n(q, \mathbf{r}) = \sum_{i=1}^{n} \log p(r_i|q, \mathbf{r}_{<i}, \theta), \tag{3}$$

where S_n is used to rank candidate *docid* sub-sequences. In GDR, a beam search creates a ranked list of candidate *docid* sequences. This method iteratively keeps the top N candidate *docid* sub-sequences of length n, extends each sub-sequence to the next position by sampling multiple token candidates for each *docid* candidate according to Equation (2), and then selects the top N candidates again based on their S_{n+1} scores.

The hierarchical structure of *docid* in GDR allows for a smaller vocabulary and shorter content than traditional seq2seq tasks. *docid* tokens in different positions carry distinct semantics despite having the same embeddings. For example, $\mathbf{r} = [1, 4]$ and $\mathbf{r}' = [2, 4]$ share the token 4, but represent different document sub-clusters. Unlike natural language, where tokens generally have consistent meanings, GDR's position-specific semantics necessitate separate embedding matrices W_i for different positions as defined in Equation (2), unlike the unified W used in natural language tasks.

4 Bi-Level Early Exiting for Generative Document Retrieval

We introduce *Bi-Level Early Exiting for Generative Document Retrieval (BiLEE)*, designed to expedite GDR using the *Layer Level Early Exiting (LLEE)* and *Token Level Early Exiting (TLEE)* strategies. Specifically, LLEE dynamical determines the optimal layer for an early exit and calibrates token-specific thresholds within a revised learn-then-test (LTT) framework to minimize unnecessary computations. Meanwhile, TLEE curtails low-confidence beam search branches, effectively narrowing the search space by omitting tokens irrelevant to the query.

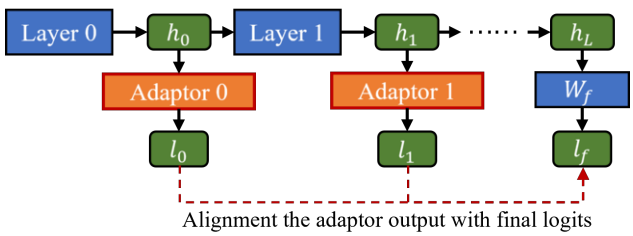


Figure 2. Adaptor training scheme. The adaptor is trained to transform earlier layer hidden states into the final layer outputs. The transformer layers of the GDR model are frozen during adaptor training.

4.1 Layer Level Early Exiting (LLEE)

First, LLEE dynamically adjusts computing depth by determining how many decoder layers to skip during inference to conserve computational resources. It calculates the confidence score c_j^i for layer j at token i, with a corresponding exit threshold λ_j^i. The framework exits at layer j_e when e represents the optimal exit position.

$$\begin{aligned} j_e &= \min(\{j \mid c_j^i > \lambda_j^i\}), \\ M_{early}(q, r_{<i}) &= \text{SoftMax}(W^{j_e} h_i^{j_e}), \end{aligned} \tag{4}$$

where W^{j_e} represents the output embedding matrix of exited layer j_e which projects the hidden states $h_i^{j_e}$ to the final logits.

Previous early exiting approaches for LLMs in natural language tasks [28] involve sharing the embedding matrix W^{j_e} across all layers, including the final word embedding, and fine-tuning the entire model using a cross-entropy loss function. However, since *docid* in GDR consists of a much smaller vocabulary size $(10 \sim 300)$ than typical sequence generation models (> 30000), naively applying such an approach leads to significant overfitting such that the GDR model converges to a performance level much below that of the original model. Furthermore, prior works set the exit threshold λ_l^i by a fixed exponential function that monotonically decreases with token position, which does not suit GDR's variant token difficulty nature.[2]

We developed LLEE to manage varying token difficulties and mitigate overfitting dynamically. Rather than fine-tuning the entire model, we implement layer-wise adaptors A_i with shared weights and unique biases to reflect varying confidence levels across layers while keeping the trained GDR model parameters fixed. Confidence scores c are derived from the difference between the top two outputs of the adaptor, as calculated by $\text{SoftMax}(A^l h_i^l)$.

As illustrated in Figure 2, our adaptor training objective is to minimize the KL divergence between the final logits and the early skipped logits, with the aim of ensuring that the output of the probability distribution at each exit layer closely approximates the probability distribution of the final result. Namely,

$$\mathcal{L}_{gt} = \sum_{j=1}^{L} \omega^j \mathcal{L}_{KL}\left(A^j h^j, W^L h^L\right). \tag{5}$$

We average losses for each layer to obtain the objective function. The ω^j is each layer's layer weights, and we employ a reversed layer weighting strategy $\omega^j = (L-j)/\sum_{k=1}^{L} k$ to favor lower layers.

For exit thresholds λ, we use a token-wise learn-then-test calibration algorithm to determine the optimal value for each token position. The learn-then-test framework redefines hyper-parameter optimization as a multiple-testing procedure. Employing a textual consistency calibrating framework and an exit threshold grid $\Lambda = (\lambda_1, \ldots, \lambda_T)$, the LTT calibration hypothesis is established.

$$\mathcal{H} : \left(\mathbb{E}\left[\mathcal{D}\left(\mathbf{M}_{early}\left(S_t, \Lambda\right), \mathbf{M}_{full}\left(S_t\right)\right)\right] \geq \delta\right) \tag{6}$$

where $\mathcal{D}$ is the retrieval result dissimilarity function, a function that can measure the difference in retrieval performance between two models. $\delta \in (0, 1)$ stands for error tolerance between the performance of full and early exited models. Hypothesis $\mathcal{H}$ can be rejected by the Hoeffding-Bentkus inequality p-value p_{HB}[9].[3] Considering the cascading errors brought about by the hierarchical identifiers in GDR, *i.e.*, errors in preceding tokens potentially causing the entire sequence to be incorrect, we can iteratively calibrate these thresholds token-by-token, allowing them to converge to an optimal configuration. Therefore, we proposed the token-wise calibrating algorithm with step size d in Algorithm 1. Note that even we did not fine-tune the threshold for each layer, each layer's adaptor possesses its own bias to reflect the confidence level for exiting at different layers.

Algorithm 1 Token-wise threshold calibrating algorithm

```
function CALIBRATE((M_early, M_full, S_t, δ, ϵ)
    Λ = [λ_1, λ_2, ··· , λ_T] = [1.0, 1.0, ··· , 1.0]
    F = True
    while F do
        F = False
        for i = 1 to T do
            Randomly sample n elements from S_t to form s.
            λ_i = λ_i − d
            Ê = D(M_early(s, Λ), M_full(s))
            p_HB = Hoeffding-Bentkus(Ê, δ, n,)
            if p_HB > ϵ then
                F = True
            else
                λ_i = λ_i + d
    return Λ
```

4.2 Token Level Early Exiting (TLEE)

While GDR generates a *docid* based on a query, beam search simultaneously allows multiple candidate *docid* sequences. Beam search selects the top N candidates at each position, corresponding to each level in the document cluster hierarchy. Typically, N remains constant as a hyperparameter, ensuring consistent candidate exploration. However, extensive ranked content is necessary for applications like web search, leading to a challenging trade-off to consider. Specifically, setting N too small limits the possible beam search exploration, thus degrading the performance; setting N too large causes the search space and, thereby, the computation complexity to explode.

To tackle this challenge, we aim to design a novel early exiting strategy for the beam search process. We notice that due to the cascading error characteristics of GDR, even with a large beam size, the diversity in search results for the initial tokens remains extremely low. For example, over 94% of the correct first tokens appear within the top 5 candidates. Therefore, we introduce **Token Level Early Exiting** to efficiently minimize computational overhead by rapidly abandoning unpromising paths. This is achieved by combining the above-mentioned LTT framework with a token-level early exit strategy. Specifically, after computing each token, any beam hypothesis

[2] Refer to section 5.3.3 for qualitative proof of variant token complexity.

[3] This p-value can be obtained by combining inverted Hoeffding's inequality [9] and Bentkus inequality [1].

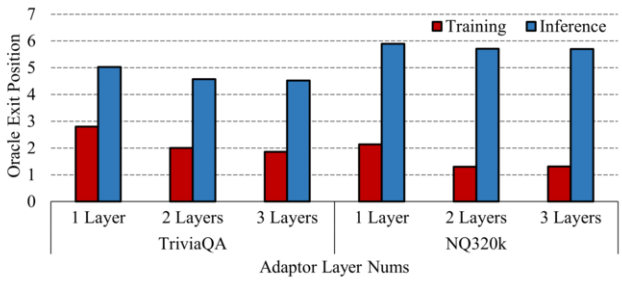


Figure 3. Comparison of the Oracle Exit Positions. During inference, the Oracle Exit Positions are 2 to 4 behind the training case, indicating a serious Training-Inference Misalignment.

with sequence or token confidence below a predefined threshold is discarded, thereby shortening the candidate list. The LTT framework adjusts the token confidence threshold λ_t similar to the LLEE threshold to maintain the integrity of the original model's performance.

Note that TLEE stands apart from earlier beam pruning techniques [7]. Unlike traditional methods, TLEE dynamically adjusts thresholds for each token, catering to the variable search complexities across different tokens in GDR. Furthermore, it uses absolute thresholds instead of relative ones, reducing uncertainty during the LTT phase and thus streamlining the complexity of the ranking process within the LTT framework.

4.3 Training-Inference Misalignment

In previous research on early exiting for generation models, such as SkipDecode [6] and CALM [28], parameters of early exiting components are typically trained using Teacher-Forcing mode. During Teacher-Forcing training, adaptors process hidden states h generated from ground truth tokens as depicted in Equation 1, $r = [r_1, r_2, \ldots,]$, tokens following the first one will only encounter the correct preceding token and their corresponding hidden states. However, during inference, adaptors or or other corresponding early exiting components frequently handle incorrect previous tokens, leading to errors in result and confidence estimation, a discrepancy known as **Training-Inference Misalignment**.

First, we introduce *Oracle Exit Position*, which is the point where the first adaptor produces the same prediction result as the full model, *i.e.*,

$$j_o = \min\left(\{j_o \mid \text{argmax}(W^{j_o} h_j^{j_o}) = \text{argmax}(W^L h^L)\}\right). \quad (7)$$

This metric assesses the effectiveness of an early exiting system by pinpointing the earliest possible point for an error-free exit. As illustrated in Figure 3, the Oracle Exit Position is notably earlier during the training phase (*i.e.*, under Teacher-Forcing) compared to the inference phase. For example, in the NQ320k experiment using a 2-layer adaptor, the model could correctly exit after an average of 1.5 layers during training. In contrast, 5.7 layers were required to achieve a correct exiting point during inference. Improving adaptor capabilities can address this discrepancy, though at the expense of increased computational demand. This issue was deemed minor in CALM, likely due to smaller beam sizes in traditional generation tasks with early exiting, which reduces the impact of misalignment compared to GDR.

To mitigate this issue, we refined the training approach, which entails training with the layer hidden states and final hidden states acquired during beam search. This is expressed as follows.

$$\begin{aligned} \mathcal{L}_{gen} &= \sum_{i=1}^{T}\sum_{j=1}^{L} \omega_i \mathcal{L}_{KL}\left(W_i^j h_i^j, W_i^L h_i^L\right) \\ \omega_i &= \max\left(\text{SoftMax}\left(W_i^L h_i^L\right)\right) \\ h_i &\in \text{Beam Search}\left(\text{Encoder}\left(q\right)\right) \end{aligned} \quad (8)$$

ω_i represents the confidence weight designed to prevent hypotheses with excessively low confidence from disrupting learning. T and L denote the maximum number of tokens and layers, respectively. h_i refers to all hidden states generated in the beam search at token position i.

5 Experiments

5.1 Experiment Setting

We experiment on two popular question-answering databases with natural language queries and corresponding datasets: TriviaQA [11], consisting of 78k query-document pairs, and NQ320k [14], consisting of 320k query-document pairs. We employ the predefined training and validation splits for evaluation. Following previous works by Wang et al. [33] and Zhuang et al. [39], we use a T5-Large[23] LLM with a 24-layer encoder and a 12-layer decoder and designate a 2-layer multilayer perceptron with layer normalization and a dropout layer as the adapter. For model training, queries are generated using DocT5Query, and the first 64 terms of each document are also utilized as queries. The training batch size is set to 256, the inference batch size to 16, and the learning rate to 1×10^{-4}, with an Adam optimizer. We evaluate GDR performance using the standard document retrieval metrics Recall@N and Mean Reciprocal Rank (MRR).[4] All experiments are performed using Python 3.9.12, PyTorch 2.0.1, and HuggingFace Transformers 3.4.0, and are run on an HP DL580 server with an Intel 2.10GHz CPU, 1TB RAM, and an NVIDIA V100 GPU.

5.2 Main Result

We first conduct a quantitative analysis comparing our BiLEE with the leading Encoder-Decoder early exiting approach CALM [28], and the unaccelerated original GDR model [33]. We evaluate various beam sizes from 5 to 100 and document the outcomes for each size. As detailed in Table 1, BiLEE significantly surpasses CALM in performance across multiple beam sizes and datasets, achieving up to a $12.59\times$ reduction in FLOPs and a $2.53\times$ increase in inference speed for the NQ320k dataset at a 100 beam size. For the TriviaQA dataset, our method also shows considerable gains, with a $4.34\times$ reduction in FLOPs and a $1.92\times$ improvement in throughput, underscoring its effectiveness.[5]

On the TriviaQA dataset, our method enables earlier exits by an average of two layers compared to CALM, with a lesser impact on recall, offering a substantial performance benefit. For NQ320k, a more complex dataset compared with TriviaQA, CALM achieved $1.60\times$

[4] MRR measures the inverse rank of the first relevant document retrieved [32], while Recall@N measures how often the desired document appears in the top N results.

[5] It's worth noting that the improvements in throughput are less marked than those in FLOPs, potentially due to factors like CPU-GPU communication or scheduling bottlenecks, suggesting areas for further throughput optimization.

Table 1. Main Results

	Beam Size	Recall at 1	Recall at 5	Recall at 10	Recall at 20	Recall at 50	Recall at 100	MRR	Exit At	TP	FLOPs	Acc. Rate TP	Acc. Rate FLOPs
						Baseline							
NQ320k	**5**	65.41%	81.44%	/	/	/	/	0.718	/	49.72	1.27E+11	1.0x	1.0x
	10	65.86%	81.84%	85.19%	/	/	/	0.723	/	27.38	2.36E+11	1.0x	1.0x
	20	65.86%	81.84%	85.17%	88.01%	/	/	0.725	/	15.71	4.68E+11	1.0x	1.0x
	50	65.86%	81.84%	85.17%	88.05%	90.56%	/	0.726	/	6.40	1.2E+12	1.0x	1.0x
	100	65.86%	81.84%	85.17%	88.05%	90.56%	92.23%	0.726	/	3.22	2.41E+12	1.0x	1.0x
						CALM							
	5	42.16%	61.07%	/	/	/	/	0.497	8.94	47.09	9.37E+10	0.95x	1.35x
	10	42.25%	62.91%	68.28%	/	/	/	0.51	9.10	30.77	1.67E+11	1.12x	1.41x
	20	42.25%	63.07%	68.94%	73.60%	/	/	0.514	9.22	18.58	3.22E+11	1.18x	1.45x
	50	42.25%	63.07%	68.94%	73.75%	78.75%	/	0.516	9.20	8.17	7.93E+11	1.28x	1.51x
	100	42.25%	63.07%	68.94%	73.75%	78.66%	82.01%	0.516	8.80	4.33	1.51E+12	1.34x	1.60x
						BiLEE							
	5	65.40%	80.59%	/	/	/	/	0.715	6.84	54.43	6.15E+10	1.09x	2.06x
	10	65.48%	80.55%	83.92%	/	/	/	0.718	6.90	38.26	8.91E+10	1.40x	2.64x
	20	65.48%	80.13%	83.97%	86.25%	/	/	0.719	6.78	26.51	1.23E+11	1.69x	3.81x
	50	65.48%	79.74%	83.59%	86.28%	89.03%	/	0.719	6.50	14.04	1.65E+11	2.20x	7.24x
	100	65.48%	79.72%	83.51%	86.17%	89.03%	90.42%	0.719	6.36	8.16	1.91E+11	2.53x	12.59x
						Baseline							
TriviaQA	**5**	73.29%	89.31%	/	/	/	/	0.38	/	42.00	1.22E+11	1.0x	1.0x
	10	73.30%	90.01%	92.29%	/	/	/	0.384	/	26.93	2.21E+11	1.0x	1.0x
	20	73.30%	90.04%	92.61%	93.96%	/	/	0.385	/	15.90	4.22E+11	1.0x	1.0x
	50	73.32%	90.04%	92.61%	94.08%	95.48%	/	0.385	/	7.08	1.04E+12	1.0x	1.0x
	100	73.30%	90.04%	92.61%	94.08%	95.46%	96.44%	0.385	/	3.58	2.12E+12	1.0x	1.0x
						CALM							
	5	67.02%	84.86%	/	/	/	/	0.348	8.76	49.20	9.16E+10	1.17x	1.34x
	10	67.07%	86.26%	89.11%	/	/	/	0.353	8.77	32.24	1.59E+11	1.20x	1.39x
	20	67.07%	86.29%	89.79%	91.69%	/	/	0.354	8.74	19.95	2.91E+11	1.25x	1.45x
	50	67.07%	86.29%	89.79%	91.69%	94.04%	/	0.355	8.75	8.87	6.91E+11	1.25x	1.51x
	100	67.07%	86.29%	89.81%	91.91%	94.17%	95.16%	0.355	8.52	4.51	1.31E+12	1.26x	1.62x
						BiLEE							
	5	72.41%	87.91%	/	/	/	/	0.374	6.55	53.90	6.48E+10	1.28x	1.89x
	10	72.42%	88.50%	90.88%	/	/	/	0.377	6.73	36.70	1.04E+11	1.36x	2.13x
	20	72.42%	88.53%	91.18%	92.97%	/	/	0.378	6.87	24.03	1.76E+11	1.51x	2.40x
	50	72.42%	88.53%	91.18%	93.07%	94.79%	/	0.378	6.91	12.16	3.27E+11	1.72x	3.19x
	100	72.42%	88.53%	91.18%	93.07%	94.77%	95.69%	0.378	6.79	6.89	4.89E+11	1.92x	4.34x

Table 2. Ablation experiment results

	Exp. Name	Recall Rate	LLEE Exit At	MRR	Acc.Rate TP	Acc.Rate FLOPs
NQ320k	Baseline	92.23%	/	0.726	1.00x	1.00x
	+ LLEE	90.82%	7.09	0.714	1.34x	1.94x
	+ TLEE	91.66%	/	0.726	1.74x	2.95x
	+ LLEE + TLEE	90.42%	6.36	0.719	2.53x	12.59x
TriviaQA	Baseline	96.44%	/	0.385	1.00x	1.00x
	+ LLEE	95.88%	6.99	0.379	1.46x	2.00x
	+ TLEE	96.30%	/	0.385	1.45x	1.92x
	+ LLEE + TLEE	95.69%	6.79	0.378	1.92x	4.34x

acceleration in FLOPS but suffered a notable 10.22% drop in recall rate, indicating that it is failing to achieve satisfactory accuracy on complex datasets. This is particularly evident from the significant decline in the MRR metric, highlighting its reduced ranking accuracy. Moreover, these findings indicate that CALM's fine-tune-based approach may not be suited for the GDR method due to overfitting issues related to a limited vocabulary, leading to premature exits and a cascade of errors. Additionally, our method demonstrates strong acceleration effects even with smaller beam sizes, achieving a $1.89\times$ to $2.06\times$ increase in FLOPS savings and a $1.09\times$ to $1.28\times$ boost in throughput. Notably, our method performs better on the NQ320k dataset, likely due to its larger data volume and longer average *docid* length, which enlarged TLEE's ability to reduce computational wastage.

5.3 In-depth Analysis

We present the in-depth analysis, including an ablation study that evaluates the individual contributions of LLEE and TLEE, a comparative examination of different adaptor types, and an analysis of layer-level exit positions and token-level exiting rates during the operation of BiLEE.

5.3.1 Ablation Study

We carried out an ablation study to evaluate the efficacy of various acceleration techniques, comparing the performance across the original model, using only TLEE or LLEE, and a final method employing both acceleration methods. Due to the space limitation, we only present results with beam size 100. Unless otherwise noted, this beam size setting will also be applied in subsequent experiments. As

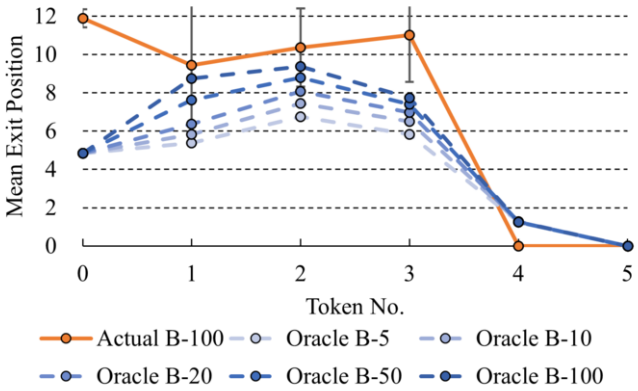


Figure 4. Mean exit positions for NQ320k while inference. The dashed line represents the Oracle Exit Position, and the B-N stands for beam size N. Both actual and Oracle show non-monotonic trends, with the Oracle Exit Position shifting later as beam size increases.

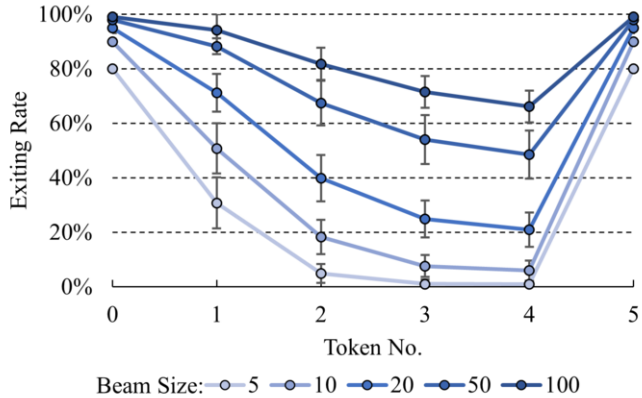


Figure 5. Token level early exiting rate for NQ320k. Aside from the final token, earlier tokens are more frequently early exited, which aligns with our observation that earlier tokens exhibit less diversity.

Table 3. Comparison of adaptor design types

	Share	Params	Recall Rate	MRR	Acc. Rate TP	Acc. Rate FLOPs
NQ320k	Weights & bias	958k	90.42%	0.719	2.53x	12.59x
	Only Weights	943k	90.37%	0.717	2.22x	11.92x
	No Sharing	11.5m	90.80%	0.719	2.21x	11.54x
TriviaQA	Weights & bias	958k	95.69%	0.378	1.92x	4.34x
	Only Weights	943k	95.21%	0.377	1.84x	3.95x
	No Sharing	11.5m	95.60%	0.377	1.92x	4.72x

demonstrated in Table 2, each acceleration method can effectively speed up inference, and their combined effect surpasses the sum of the two. This is due to the TLEE reducing the number of branches with low confidence, which increases the proportion of branches that the LLEE can prematurely exit. The advancement of the LLTT exit point further corroborates this observation.

5.3.2 *Adaptor Design*

In order to identify an efficient adaptor design, we tested several different types of adaptors, including one where all weights and biases are shared (*i.e.*, all layers sharing one same adaptor), one where only weights are shared (the approach we utilized), and one where each layer has an independent adaptor. It can be observed from Table 3 that sharing all parameters, due to the lack of differentiated biases reflecting the confidence levels of each layer, results in a loss of accuracy and acceleration efficiency. On the other hand, using an independent adaptor for each layer significantly increases the number of parameters. Although this approach can improve acceleration performance in certain cases, we consider it an alternative option when sensitivity to parameter count is not a concern.

5.3.3 *Layer Level Exit Position*

To analyze the differences in layer level early exiting among various tokens, we measure the mean exit positions of each token during inference. As illustrated in Figure 4, the initial token does not typically trigger an early exit in our model due to its significant impact on subsequent sequences. In subsequent tokens, our model tends to exit early at the second token and then progressively adopts a more cautious strategy until reaching the EOS tokens, which can exit immediately. The phenomenon observed may be attributed to the second token being situated at the intersection of semantics diversity and cascading error impacts. Specifically, before the second token, the main barrier to early exits is the cascading errors resulting from incorrect predictions. After the second token, the increase in the semantics diversity in the search space makes it harder for the adaptor to mimic the full model's output with enough confidence to exit. The above observations validate our hypothesis that the GDR model exhibits significant differences in the difficulty of early exits across different token positions.

5.3.4 *Token Level Exiting Rate*

We explored our token-level early exiting algorithm across different beam sizes and token positions. As shown in Figure 5, the proportion of early exits is higher for earlier tokens, supporting our hypothesis that the initial tokens generally exhibit low diversity. An exception is the final token, where hypotheses other than the End-Of-Sentences token typically receive very low confidence scores and cause a higher exiting rate. An extensive search on these tokens consumes significant computational resources unnecessarily. Moreover, with larger beam sizes, there is a marked increase in the exiting rate due to a larger beam size generating more low-confidence branches. Those patterns indicate that our algorithm, calibrated with the LTT, effectively exits at appropriate token-wise thresholds.

6 Conclusion

In this paper, we introduced the Bi-Level Early Exiting for Generative Document Retrieval (BiLEE), a novel framework employing bi-level early exiting strategies comprising Layer Level Early Exiting (LLEE) and Token Level Early Exiting (TLEE). Our approach innovatively adjusts the early exiting thresholds dynamically, significantly enhancing computational efficiency in GDR without compromising accuracy. By integrating LLEE to manage layer-specific exit decisions and TLEE to refine token-level search space efficiency during the beam search process, BiLEE significantly reduces FLOPs consumption and improves throughput in GDR tasks.

Acknowledgements

This work was supported by the NSTC, Taiwan, and MOE, Taiwan, under Grant NSTC 113-2223-E-002-011, NSTC 111-2221-E-002-135-MY3, and MOE 113L9009.

References

[1] V. Bentkus. On hoeffding's inequalities. *Annals of probability*, pages 1650–1673, 2004.

[2] M. Bevilacqua, G. Ottaviano, P. Lewis, S. Yih, S. Riedel, and F. Petroni. Autoregressive search engines: Generating substrings as document identifiers. *Advances in Neural Information Processing Systems*, 35: 31668–31683, 2022.

[3] H. Chen, X. Liu, D. Yin, and J. Tang. A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter*, 19(2):25–35, 2017.

[4] Y. Chen, X. Pan, Y. Li, B. Ding, and J. Zhou. Ee-llm: Large-scale training and inference of early-exit large language models with 3d parallelism. *arXiv preprint arXiv:2312.04916*, 2023.

[5] N. De Cao, G. Izacard, S. Riedel, and F. Petroni. Autoregressive entity retrieval. In *ICLR 2021-9th International Conference on Learning Representations*, volume 2021. ICLR, 2020.

[6] L. Del Corro, A. Del Giorno, S. Agarwal, B. Yu, A. Awadallah, and S. Mukherjee. Skipdecode: Autoregressive skip decoding with batching and caching for efficient llm inference. *arXiv preprint arXiv:2307.02628*, 2023.

[7] M. Freitag and Y. Al-Onaizan. Beam search strategies for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*. Association for Computational Linguistics, 2017.

[8] X. Gao, W. Zhu, J. Gao, and C. Yin. F-pabee: flexible-patience-based early exiting for single-label and multi-label text classification tasks. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[9] W. Hoeffding. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding*, pages 409–426, 1994.

[10] S. Hofstätter, N. Craswell, B. Mitra, H. Zamani, and A. Hanbury. Are we there yet? a decision framework for replacing term based retrieval with dense retrieval systems. *arXiv preprint arXiv:2206.12993*, 2022.

[11] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer. 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)*, pages 1601–1611, 2017.

[12] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W.-t. Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, 2020.

[13] J. D. M.-W. C. Kenton and L. K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

[14] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019.

[15] Y. Li, N. Yang, L. Wang, F. Wei, and W. Li. Multiview identifiers enhanced generative retrieval. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6636–6648, 2023.

[16] W. Liu, P. Zhou, Z. Wang, Z. Zhao, H. Deng, and Q. Ju. Fastbert: a self-distilling bert with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6035–6044, 2020.

[17] B. P. Lowerre and B. R. Reddy. Harpy, a connected speech recognition system. *The Journal of the Acoustical Society of America*, 59(S1):S97–S97, 1976.

[18] Y. Matsubara, M. Levorato, and F. Restuccia. Split computing and early exiting for deep learning applications: Survey and research challenges. *ACM Computing Surveys*, 55(5):1–30, 2022.

[19] B. Mitra, F. Diaz, and N. Craswell. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th international conference on world wide web*, pages 1291–1299, 2017.

[20] G. Navarro. Spaces, trees, and colors: The algorithmic landscape of document retrieval on sequences. *ACM Computing Surveys (CSUR)*, 46 (4):1–47, 2014.

[21] T. Nguyen and A. Yates. Generative retrieval as dense retrieval. *arXiv preprint arXiv:2306.11397*, 2023.

[22] R. Pradeep, K. Hui, J. Gupta, A. Lelkes, H. Zhuang, J. Lin, D. Metzler, and V. Tran. How does generative retrieval scale to millions of passages? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1305–1321, 2023.

[23] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

[24] R. Ren, Y. Qu, J. Liu, W. X. Zhao, Q. She, H. Wu, H. Wang, and J.-R. Wen. Rocketqav2: A joint training method for dense passage retrieval and passage re-ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2825–2835, 2021.

[25] S. E. Robertson and S. Walker. On relevance weights with little relevance information. In *Proceedings of the 20th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 16–24, 1997.

[26] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford, et al. Okapi at trec-3. *Nist Special Publication Sp*, 109:109, 1995.

[27] G. Salton, A. Wong, and C.-S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[28] T. Schuster, A. Fisch, J. Gupta, M. Dehghani, D. Bahri, V. Tran, Y. Tay, and D. Metzler. Confident adaptive language modeling. *Advances in Neural Information Processing Systems*, 35:17456–17472, 2022.

[29] Y. Tay, V. Tran, M. Dehghani, J. Ni, D. Bahri, H. Mehta, Z. Qin, K. Hui, Z. Zhao, J. Gupta, et al. Transformer memory as a differentiable search index. *Advances in Neural Information Processing Systems*, 35:21831–21843, 2022.

[30] S. Teerapittayanon, B. McDanel, and H.-T. Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd international conference on pattern recognition (ICPR)*, pages 2464–2469. IEEE, 2016.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[32] E. M. Voorhees, D. M. Tice, et al. The trec-8 question answering track evaluation. In *TREC*, volume 1999, page 82, 1999.

[33] Y. Wang, Y. Hou, H. Wang, Z. Miao, S. Wu, Q. Chen, Y. Xia, C. Chi, G. Zhao, Z. Liu, et al. A neural corpus indexer for document retrieval. *Advances in Neural Information Processing Systems*, 35:25600–25614, 2022.

[34] L. Xiong, C. Xiong, Y. Li, K.-F. Tang, J. Liu, P. Bennett, J. Ahmed, and A. Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*, 2020.

[35] H. Zeng, C. Luo, B. Jin, S. M. Sarwar, T. Wei, and H. Zamani. Scalable and effective generative information retrieval. In *Proceedings of the ACM on Web Conference 2024*, pages 1441–1452, 2024.

[36] S. Zhang, Y. Liang, M. Gong, D. Jiang, and N. Duan. Multi-view document representation learning for open-domain dense retrieval. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5990–6000, 2022.

[37] W. Zhou, C. Xu, T. Ge, J. McAuley, K. Xu, and F. Wei. Bert loses patience: Fast and robust inference with early exit. *Advances in Neural Information Processing Systems*, 33:18330–18341, 2020.

[38] Y. Zhou, J. Yao, Z. Dou, L. Wu, P. Zhang, and J.-R. Wen. Ultron: An ultimate retriever on corpus with a model-based indexer. *arXiv preprint arXiv:2208.09257*, 2022.

[39] S. Zhuang, H. Ren, L. Shou, J. Pei, M. Gong, G. Zuccon, and D. Jiang. Bridging the gap between indexing and retrieval for differentiable search index with query generation. *arXiv preprint arXiv:2206.10128*, 2022.