

# Parametric Retrieval Augmented Generation

Weihang Su  
swh22@mails.tsinghua.edu.cn  
DCST, Tsinghua University  
Beijing 100084, China

Yichen Tang\*  
DCST, Tsinghua University  
Beijing 100084, China

Qingyao Ai†  
aiqy@tsinghua.edu.cn  
DCST, Tsinghua University  
Beijing 100084, China

Junxi Yan  
DCST, Tsinghua University  
Beijing 100084, China

Changyue Wang  
DCST, Tsinghua University  
Beijing 100084, China

Hongning Wang  
DCST, Tsinghua University  
Beijing 100084, China

Ziyi Ye  
DCST, Tsinghua University  
Beijing 100084, China

Yujia Zhou  
DCST, Tsinghua University  
Beijing 100084, China

Yiqun Liu  
DCST, Tsinghua University  
Beijing 100084, China

## Abstract

Retrieval-augmented generation (RAG) techniques have emerged as a promising solution to enhance the reliability of large language models (LLMs) by addressing issues like hallucinations, outdated knowledge, and domain adaptation. In particular, existing RAG methods append relevant documents retrieved from external corpus or databases to the input of LLMs to guide their generation process, which we refer to as the in-context knowledge injection method. While this approach is simple and often effective, it has inherent limitations. Firstly, increasing the context length and number of relevant documents can lead to higher computational overhead and degraded performance, especially in complex reasoning tasks. More importantly, in-context knowledge injection operates primarily at the input level, but LLMs store their internal knowledge in their parameters. This gap fundamentally limits the capacity of in-context methods. To this end, we introduce Parametric retrieval-augmented generation (Parametric RAG), a new RAG paradigm that integrates external knowledge directly into the parameters of feed-forward networks (FFN) of an LLM through document parameterization. This approach not only saves online computational costs by eliminating the need to inject multiple documents into the LLMs' input context, but also deepens the integration of external knowledge into the parametric knowledge space of the LLM. Experimental results demonstrate that Parametric RAG substantially enhances both the effectiveness and efficiency of knowledge augmentation in LLMs. Also, it can be combined with in-context RAG methods to achieve even better performance<sup>1</sup>.

## Keywords

Large Language Model, Retrieval Augmented Generation, Knowledge Representation, Parametric Information Representation

## 1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across a wide range of information retrieval (IR) and

natural language processing (NLP) tasks [5, 6, 11, 36, 48, 55]. Despite these successes, a critical limitation remains: once training is complete, an LLM's internal knowledge becomes effectively static, making it challenging to incorporate newly emerging information or knowledge not included in its pre-training data. To address this challenge, retrieval-augmented generation (RAG) has emerged as a prominent solution. RAG enables LLMs to dynamically access and utilize information beyond their pre-trained parameters by retrieving relevant information from an external corpus, thus improving their adaptability and performance [4, 12, 16, 18, 23, 44–46].

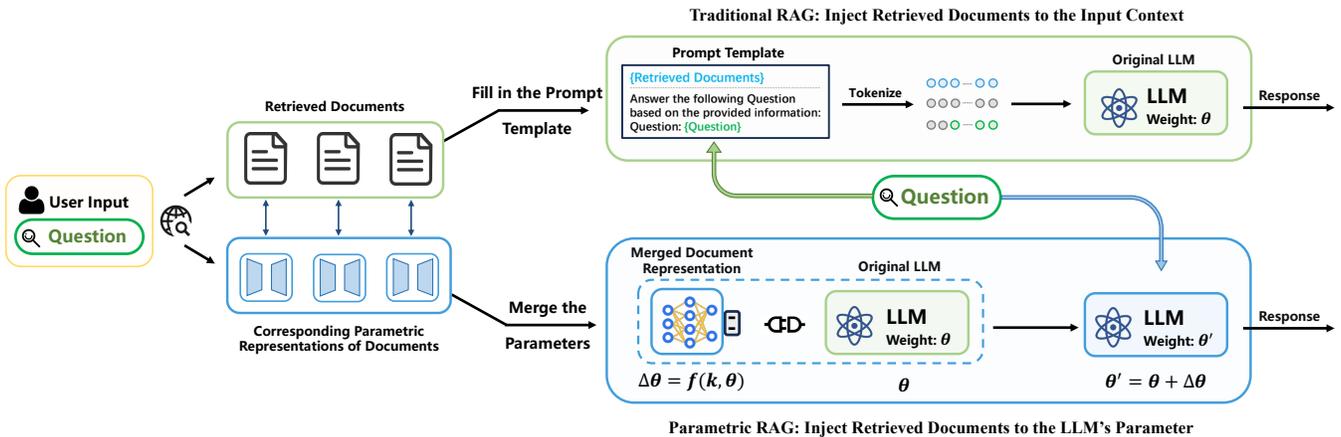
Existing studies have explored various aspects of the RAG pipeline, considering factors such as retrieval timing [2, 18, 44, 45], document selection [21, 58], and external knowledge organization [10, 15, 32]. While these innovations improve different stages of the pipeline, all RAG methods, regardless of their variations, share a common characteristic: they inject external knowledge by directly adding passages or documents into the input context of LLMs, which we refer to as the in-context knowledge injection.

Although this in-context knowledge injection approach is straightforward and often effective, recent studies have highlighted several limitations of this paradigm. First, injecting knowledge through input prompts will inevitably increase the context length. Long context not only introduces extra computational overhead and latency for LLM inference, but also hurts the performance of LLMs in understanding and utilizing external knowledge, especially in tasks that involve complex reasoning [22, 26]. Second, more importantly, the way LLMs process information in context is fundamentally different from the way they utilize internal knowledge stored in their parameters. Studies have shown that LLMs store most of their knowledge within the parameters of their neural network architecture (e.g., the parameters of their feed-forward network layers) [31, 59]. Adding passages or documents in the input context could only affect the online computation of key-value (KV) pairs in the attention networks of LLMs, but not the model's stored parameters, where its knowledge is encoded [59]. This means that LLMs may never be able to utilize external knowledge as effectively as they use their internal knowledge in in-context RAG methods. A straightforward solution to this problem is to conduct supervised fine-tuning (SFT) with retrieved documents, thereby incorporating relevant knowledge directly into the LLM's parameters. However,

\*Contributed equally

†Corresponding author

<sup>1</sup>We have open-sourced all the code, data, and models in the following anonymized GitHub link: <https://github.com/oneal2000/PRAG>



**Figure 1: An illustration of the comparison of in-context RAG and our proposed Parametric RAG paradigms: In-context RAG combines the tokens of relevant documents and the query in the input, using the original LLM  $\theta$  to answer the question without modifying its parameters. Our proposed Parametric RAG updates the LLM’s parameters  $\theta' = \theta + \Delta\theta$  based on the retrieved documents, temporarily integrating relevant knowledge into LLM’s parameters to answer the question.**

such SFT-based methods are considered suboptimal as they require substantial computational resources and GPU memory, making it impractical to inject relevant documents online for every query. In addition, they can negatively affect the original ability of the LLM to follow instructions [7, 54] and lack the flexibility of in-context methods, which allow external knowledge to be added or removed on the fly.

The observations above inspire us to raise the following research question: *Is it possible to inject external knowledge into LLM parameters effectively, efficiently, and flexibly for retrieval-augmented generation?*

To this end, we introduce a new RAG paradigm, **Parametric Retrieval Augmented Generation (Parametric RAG)**, which directly injects the external knowledge into the Feed-Forward Networks (FFN) of an LLM. Specifically, our approach begins with an offline preprocessing phase that parameterizes each document from the external corpus, transforming them into a small set of parameters (usually a few MB per document) that can be directly integrated into the downstream LLM. We refer to this set of parameters as the parametric representation of the document. In the inference phase, we conduct retrieval augmented generation following a Retrieve-Update-Generate (RUG) workflow as shown in Figure 1. The Retrieve step retrieves top- $n$  documents from the external corpus based on the input prompt following the same procedure used by the existing RAG pipeline. Then, the Update step uses the parametric representations of the retrieved documents to update the LLM. Finally, in the Generate step, we use the updated LLMs to conduct inference directly based on the original input prompt.

Theoretical and empirical analysis show that our Parametric RAG method has superior inference efficiency and outperforms state-of-the-art in-context methods on several RAG benchmarks that involve tasks with complex reasoning. While the preprocessing phase of Parametric RAG introduces an offline computational overhead, this cost is affordable and even neglectable compared to the online cost

of large-scale inference requests, leading to long-term savings in terms of power and carbon footprints. Also, similar to in-context RAG, our method can adapt to various numbers of input documents on the fly. Furthermore, our proposed Parametric RAG pipeline is in parallel with existing in-context methods. As shown in our experiments, combining our methods with in-context RAG could produce even better performance on the benchmark datasets. This indicates that parametric knowledge injection could be a fruitful direction for the future development of the RAG system.

In summary, this paper makes the following key contributions:

- We propose Parametric RAG, a new RAG paradigm that integrates external knowledge directly into LLM’s parameters.
- We propose an offline method to build parametric document representation and a Retrieve-Update-Generate pipeline to conduct Parametric RAG on the fly.
- We conduct extensive experiments to compare the state-of-the-art in-context RAG methods with our method to demonstrate the potential of Parametric RAG in terms of effectiveness and efficiency.

## 2 Related Work

Large language models have shown remarkable performance across diverse applications. However, their inherent knowledge often proves insufficient for tackling knowledge-intensive tasks, underscoring the necessity of integrating external knowledge for robust performance in these contexts. One prominent approach to address this gap is Retrieval-Augmented Generation (RAG), which improves LLMs by integrating relevant external knowledge sources [4, 8, 12, 16, 18, 23, 37, 42, 50, 51]. In the traditional RAG framework, an external retriever [9, 27, 34, 40, 41, 43, 60] or a more complex retrieval system [24, 35] retrieves relevant documents based on a query. These documents are then appended to the LLM’s input context, allowing the LLM to leverage knowledge beyond its training data [23].

Building upon the traditional RAG framework, several extensions have been proposed to improve its effectiveness and efficiency. One such extension, Adaptive RAG [17, 52], introduces adaptive retrieval strategies that actively adjust the retrieval pipeline based on the complexity of the query to improve the adaptability of RAG in different tasks. From another angle, to make in-context knowledge injection more effective in RAG scenarios, IR-CoT [49] designs prompt templates specifically tailored for RAG that demonstrate how to perform chain-of-thought (CoT) reasoning based on the given passage. Each sentence in the CoT reasoning content is then applied to retrieve more relevant documents. Another research direction, GraphRAG [10, 15, 32], employs pre-constructed knowledge graphs to retrieve graph elements that encapsulate relational knowledge relevant to the query. GraphRAG has demonstrated enhanced performance, particularly in tasks that require structured, relational information. In the context of long-form generation, where the LLM’s informational needs may change during the generation process, dynamic RAG techniques have been developed to actively decide when and what to retrieve during the generation process [3, 19, 25, 44, 45, 57]. For example, FLARE [19] triggers the retrieval module when the model’s token prediction probability falls below a predefined threshold. Similarly, DRAGIN [45] models the real-time information needs of the LLM, generating queries based on the model’s internal state and preceding context to fetch relevant external knowledge dynamically.

To summarize, existing RAG approaches have explored various aspects of the RAG pipeline, considering factors such as retrieval timing [3, 19, 25, 44, 45, 57], prompt template for in-context knowledge injection [49, 53], document selection [58], and external knowledge organization [10, 15, 32]. While these innovations improve different stages of the pipeline, all RAG methods, regardless of their variations, share a common characteristic at the knowledge injection level: relevant passages or documents are appended directly to the LLM’s input context to inject external knowledge. In contrast, our proposed Parametric RAG paradigm diverges from all the existing RAG frameworks by directly injecting documents into the LLM’s parameters. This shift in knowledge integration addresses the inherent limitations of the in-context knowledge injection methods employed in all existing RAG frameworks.

### 3 Methodology

In this section, we introduce our proposed Parametric RAG framework, shown in Figure 1. This section begins by formulating the problem and providing an overview of the Parametric RAG framework (§3.1). Next, we introduce the Offline Document Parameterization process (§3.2), which transforms documents into parametric representations through *Document Augmentation* and *Parametric Document Encoding*. Finally, we introduce the Online Inference procedure (§3.3), where the parametric representations are retrieved, merged, and integrated into the LLM to generate responses.

#### 3.1 Problem Formulation and Overview

This subsection introduces the problem formulation of the RAG task and provides an overview of our proposed Parametric RAG pipeline. Consider an LLM (denoted as  $\mathcal{L}$ ) with base parameters

$\theta$ . Given a user query  $q$ , we aim to generate an accurate response using an external corpus  $K$ . Formally, the corpus  $K$  is defined as:  $K = \{d_1, d_2, \dots, d_N\}$ , where each  $d_i$  represents a text chunk, such as documents, Wikipedia articles, or passages (for convenience, we refer to each  $d_i$  as ‘document’ in the following sections). The system contains a retrieval module  $R$  that calculates the relevance score of each document  $\{S_{d_1}, S_{d_2}, \dots, S_{d_N}\}$  corresponding to the query  $q$ . Traditional RAG paradigms select the top  $k$  documents with the highest relevance scores as relevant external knowledge and append them to the input context of the  $\mathcal{L}$ . This process is typically guided by a prompt template that instructs  $\mathcal{L}$  to generate the response based on the provided knowledge.

In contrast to the in-context RAG paradigm that injects relevant documents into the LLM’s input context, in Parametric RAG, we propose to insert documents directly into the parameters of  $\mathcal{L}$ . To achieve this, the Parametric RAG framework is designed with two stages: an offline document parameterization stage and an online inference stage with a Retrieve-Update-Generate workflow.

*Offline document Parameterization.* In this step (illustrated in Figure 2), we offline transform each document in  $K$  into a parametric representation, thereby forming a set of parameters known as the Parametric Corpus  $K_p$ . Specifically, we define:

$$K_p = \{p_i \mid p_i = f_\phi(d_i), \quad i = 1, 2, \dots, N\}, \quad (1)$$

where  $f_\phi$  is a mapping function that converts each document  $d_i$  into its corresponding parametric representation  $p_i$ . We define parametric representations  $p_i$  to possess the following properties:

- (1) The parameters  $p_i$  can be plugged into the feed-forward network weights of the LLM.
- (2) After inserting the parametric representation  $p_i$  into  $L$ , the LLM can effectively comprehend the knowledge contained within the corresponding document  $d_i$ .
- (3) Different document parameters  $p_i$  can be merged through specific algorithms, and after merging, the LLM can grasp the combined knowledge corresponding to the merged documents.

*Online Inference.* During the online inference process, our method first merges the parametric representations corresponding to the retrieved top- $k$  documents and then plugs the merged parameters into the LLM. Subsequently, the updated LLM is used to answer the user’s question. This overall framework allows for more efficient and effective knowledge injection, overcoming the limitations of traditional RAG by leveraging parameterized representations of external knowledge.

#### 3.2 Offline Document Parameterization

In this subsection, we describe the detailed process of offline parameterizing each document in the corpus  $K$  during the pre-processing phase. Given a document  $d_i$  and an LLM  $\mathcal{L}$ , our objective is to construct a parametric representation  $p_i$  of the document. This representation enables  $\mathcal{L}$  to effectively comprehend and utilize the knowledge contained in  $d_i$  during inference. To achieve this, we propose a two-step approach: *Document Augmentation* and *Parametric Document Encoding*. These steps are combined to generate robust and informative parametric representations for each document.

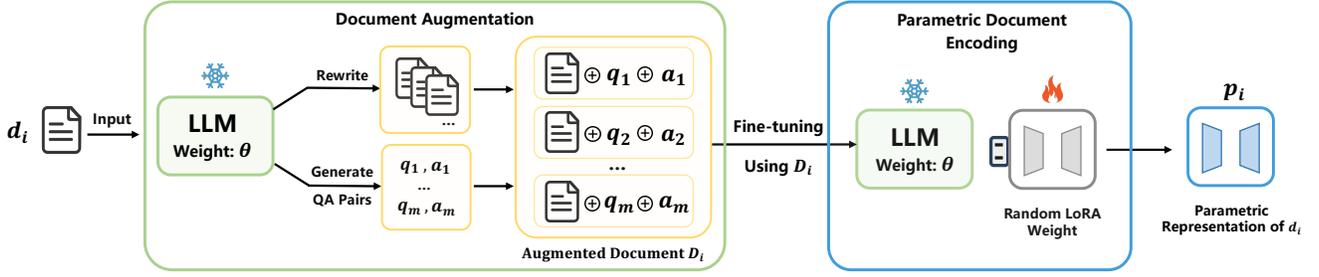


Figure 2: An illustration of how we parameterize each document  $d_i$  in the corpus during the *Offline Document Parameterization* stage.

3.2.1 *Document Augmentation*. Existing studies have shown that effectively incorporating factual knowledge from external documents into LLMs requires more than simply pre-training the model on raw text via standard next-token prediction. For example, Allen-Zhu and Li [1] find that after being trained repeatedly on a given document, LLMs can memorize its content but fail to extract and apply this knowledge effectively in downstream tasks such as question answering. To address this issue, Allen-Zhu and Li [1] propose two key strategies: (1) incorporating question-answer (QA) pairs derived from the document during training and (2) augmenting the document through multiple rewrites that express the same factual content in different forms. Their findings indicate that these two approaches enable LLMs to internalize knowledge to support accurate application in downstream tasks rather than reproducing the original text token-by-token<sup>2</sup>.

Building upon the insights discussed above, we introduce the document augmentation process consisting of two steps: **Document Rewriting and QA Pair Generation**, to construct robust and informative parametric representations for documents. Specifically, for each document, we prompt<sup>3</sup> the LLM  $\mathcal{L}$  to rewrite the content multiple times using different wording, styles, or organizational structures. Formally, each document  $d_i$  is transformed into multiple rewritten documents  $\{d_i^1, d_i^2, \dots, d_i^n\}$ , which preserve the original facts but vary in language expression. Once each document has been rewritten into multiple documents, we prompt  $\mathcal{L}$  again to generate question-answer (QA) pairs based on the original document  $d_i$ . For each document  $d_i$ ,  $\mathcal{L}$  produces a set of questions and their corresponding answers:  $\{(q_i^1, a_i^1), (q_i^2, a_i^2), \dots, (q_i^m, a_i^m)\}$ , where  $m$  is a tunable hyperparameter representing the number of QA pairs we aim to generate per document. Integrating multiple rewrites with corresponding QA pairs transforms each document  $d_i$  into a more comprehensive resource  $D_i$  that preserves its original factual content while incorporating diverse linguistic variations. Formally:

$$D_i = \{(d_i^k, q_i^j, a_i^j) \mid 1 \leq k \leq n, 1 \leq j \leq m\}, \quad (2)$$

where each  $(d_i^k, q_i^j, a_i^j)$  triple corresponds to a rewritten document  $d_i^k$  from the original document  $d_i$ , coupled with a question  $q_i^j$  and its respective answer  $a_i^j$ .

<sup>2</sup>Our experimental results corroborate these conclusions, as shown in Figure 3.

<sup>3</sup>Due to space constraints, we have not included the specific prompts in the main text. However, all prompts used in this study are available at the following anonymous link: [https://github.com/oneal2000/PRAG/blob/main/all\\_prompt.md](https://github.com/oneal2000/PRAG/blob/main/all_prompt.md)

3.2.2 *Parametric Document Encoding*. In this subsection, we introduce the second step of the offline document parameterization pipeline, where we leverage the augmented dataset  $D_i$  (defined in Eq. 2) to train the parametric representation  $p_i$  for each document  $d_i$ . Specifically, we initialize these parametric representations as low-rank matrices corresponding to the feed-forward network (FFN) parameter matrix  $W$  of the LLM  $\mathcal{L}$ , following the LoRA approach [14]. This design allows each document  $d_i$  to be associated with independently trained low-rank parameters, allowing the model to internalize the knowledge specific to  $d_i$  in a parameter-efficient manner.<sup>4</sup>

Suppose the Transformer layers in  $\mathcal{L}$  have a hidden dimension  $h$ , and the feed-forward network (FFN) within each layer has an intermediate dimension  $k$ . Consequently, each FFN layer of  $\mathcal{L}$  contains a weight matrix  $W \in \mathbb{R}^{h \times k}$ . To incorporate document-specific knowledge, we introduce low-rank matrices  $A$  and  $B$  such that

$$W' = W + \Delta W = W + AB^T, \quad (3)$$

where  $A \in \mathbb{R}^{h \times r}$  and  $B \in \mathbb{R}^{k \times r}$ , with  $r \ll \min(h, k)$ . The original weight matrix  $W$  is kept fixed, while  $A$  and  $B$  are the only trainable parameters for that layer. We denote these newly introduced parameters as  $\Delta\theta = \{A, B\}$ . By combining the pre-trained weights  $W$  and  $\Delta\theta$ , the model obtains the knowledge from the selected document. Each document in the corpus is associated with its instance of  $\Delta\theta$ .

For each document  $d_i$ , we train its corresponding parametric representation  $\Delta\theta$  using its corresponding augmented dataset  $D_i$ . Recall from Eq. 2 that  $D_i$  contains triplets  $(d_i^k, q_i^j, a_i^j)$ . For each triplet, we concatenate  $d_i^k$ ,  $q_i^j$ , and  $a_i^j$  into a token sequence:

$$x = [d_i^k \oplus q_i^j \oplus a_i^j], \quad (4)$$

where  $[\cdot \oplus \cdot]$  indicates concatenation. Let  $T$  be the total number of tokens in  $x$ . We adopt a standard sequential language modeling objective to ensure that the LLM internalizes knowledge from the entire augmented text (i.e., both the documents and QA pairs). Specifically, we optimize:

$$\min_{\Delta\theta} \sum_{(d_i^k, q_i^j, a_i^j) \in D_i} \sum_{t=1}^T -\log P_{\theta+\Delta\theta}(x_t | x_{<t}), \quad (5)$$

<sup>4</sup>Other parameter-efficient methods (e.g., Adapters or Prefix-Tuning) could also be used; exploring them is left for future work. In this work, we chose LoRA because it offers practical advantages over other alternatives. For example, LoRA is easier to merge compared to Adapters, and it requires less computational overhead during inference compared to Prefix-Tuning.

where  $\theta$  are the frozen pretrained parameters of the LLM, and  $\Delta\theta = \{A, B\}$  are the trainable low-rank matrices introduced in Eq. 3. The innermost summation is taken over all tokens  $x_t$  in the concatenated input sequence (document, question, and answer)<sup>5</sup>. This design inherently encourages the LLM to internalize the factual details in the documents into its parameters during training. Although the generated question-answer pairs do not directly cover all the facts within the document, repeated training on the documents’ tokens allows the model to reinforce its understanding of the textual content. Consequently, once the training is complete, the parametric representation  $\Delta\theta$  serves as a lightweight document-specific knowledge representation that can be directly added to the original model  $\mathcal{L}$  at inference time.

Notably, this entire process can be conducted offline, as each document (or batch of documents) is processed to produce its respective low-rank representation  $\Delta\theta$ . At inference time, we only need to load the LoRA parameters corresponding to the specific document(s) rather than appending the document directly into the LLM’s context. The computational cost of loading these parametric representations constitutes only a minimal portion, approximately 1% of the computation required to decode a single token.

**3.2.3 Discussion on LoRA Initialization.** In our proposed training framework, the LoRA parameters for each document  $d_i$  are initialized randomly without any warm-up stage. This choice is deliberate and aligns with our goal of developing a general-purpose parametric knowledge injection method rather than one tailored to specific downstream tasks. If not explicitly mentioned otherwise, we initialize LoRA with random values. However, randomized LoRA initialization is not necessarily the most effective way to train parametric document representations. For example, we could pre-train the random LoRA with a couple of few-shot examples following the same method described with Eq. (5) and save the LoRA weight  $W_{warm-up}$  to initialize the training of each document’s LoRA (i.e., the document’s parametric representation). Our experiment (Section § 5.2) demonstrates that this warm-up process can significantly improve the performance compared to random initialization on RAG tasks, indicating that a *task-aware* initialization can further enhance the effectiveness of parametric knowledge injection for specific downstream tasks. Yet, we use random initialization for LoRA if not mentioned explicitly for simplicity and broad applicability across various tasks.

### 3.3 Online Inference

In the previous stage (§3.2.2), we generated a set of document-specific low-rank parameters for each document in the corpus  $K$ . This section describes how these parameters are utilized for RAG pipelines. Given a user query  $q$ , our proposed Parametric RAG pipeline proceeds through three steps: **Retrieve**, **Update**, and **Generate**. The following section details each step and illustrates the underlying mathematical operations.

**3.3.1 Retrieve.** We first use a retriever  $R$  to calculate a relevance score  $S_{d_i}$  for each document  $d_i \in K$  to the query  $q$ . We then select the top- $k$  documents with the highest relevance scores, denoted

as  $\{d_1, d_2, \dots, d_k\} \subseteq K$  as the relevant external knowledge. Each retrieved document  $d_i$  has a corresponding parametric representation, i.e., a pair of low-rank matrices  $(A_i, B_i)$ , previously obtained by the procedure described in §3.2.2.

**3.3.2 Update.** After retrieval, we merge the low-rank matrices from the top- $k$  retrieved documents to form a single plug-in module for the LLM. Following the setting of LoRA [14] convention, we use a scalar scaling factor  $\alpha$  to modulate the final update. The merged weight update,  $\Delta W_{\text{merge}}$ , is computed by summing over all retrieved documents:

$$\Delta W_{\text{merge}} = \alpha \cdot \sum_{j=1}^k A_j B_j^T. \quad (6)$$

Intuitively,  $\Delta W_{\text{merge}}$  combines the knowledge from multiple relevant documents into a single low-rank update that can be applied to the LLM’s base parameters. Once we obtain  $\Delta W_{\text{merge}}$ , we update the original feed-forward weight  $W$  by:  $W' = W + \Delta W_{\text{merge}}$ , thus yielding the final set of parameters for that layer at inference time. Conceptually,  $W'$  encodes the base model’s knowledge plus the aggregated knowledge from the top- $k$  retrieved documents.

**3.3.3 Generate.** After updating all feed-forward layers in the Transformer with  $\Delta W_{\text{merge}}$ , we obtain a temporary model  $\mathcal{L}'(\theta')$ , where  $\theta'$  represents the updated model parameters, which are obtained by incorporating the merged low-rank parameters for all retrieved documents. We can then directly use  $\mathcal{L}'$  to generate the final response to the query  $q$  using a standard left-to-right decoding process.

### 3.4 Discussion on Time/Space Efficiency

**3.4.1 Computation Cost.** The computation cost of our method can be divided into offline preprocessing cost and online inference cost. The offline cost primarily arises from the *Parametric Document Encoding* (§3.2.2). Let  $|d|$  be the average number of tokens in a document  $d$ , and  $h$  be the hidden dimension size of the LLM. The computational complexity of a typical decoder-only LLM is  $O(|d|^2 h + |d| h^2)$ , where the attention layers complexity is  $O(|d|^2 h)$  plus the FFN layers  $O(|d| h^2)$ . Theoretically, our method only introduces a constant coefficient change on the number of tokens processed, thus the overall time complexity remains  $O(|d|^2 h + |d| h^2)$ . Based on our implementation settings detailed in § 4.3, the Data Augmentation process takes the original document  $d$  as input and subsequently generates approximately  $2|d|$  new tokens. This process requires computational costs equivalent to a forward pass over  $3|d|$  tokens, including the decoding of  $1|d|$  tokens and the inference of  $2|d|$  tokens. Training LoRA parameters on these augmented tokens requires a forward pass over  $3|d|$  tokens and a backward pass equivalent to processing  $6|d|$  tokens (typically about twice the forward-pass cost), resulting in an overall computational cost equivalent to processing  $9|d|$  tokens. Adding the  $3|d|$  tokens from the Document Augmentation phase, the total computational cost is approximately the cost of decoding  $12|d|$  tokens in the LLM.

The online inference cost mainly depends on the number of input and output tokens. For simplicity, we focus on input tokens and ignore the variance in output tokens since they vary significantly from tasks and LLMs. Let  $|q|$  represent the length of input prompt/question  $q$ , and  $t$  be the number of retrieved documents.

<sup>5</sup>The loss is computed not only on the answer but also across the entire concatenated sequence, including the documents and the question

Since the time needed to load the LoRA parameters for  $t$  documents is neglectable, the inference time complexity of our method is  $O(|q|^2h + |q|h^2)$ . In contrast, the time complexity of in-context RAG methods is  $O((t|d| + |q|)^2h + (t|d| + |q|)h^2)$ , which means that our method can save  $O(t^2|d|^2h + t|d||q|h + t|d|h^2)$  time for online inference. Empirically, suppose that the lengths of  $q$  and  $d$  are approximately the same and significantly smaller than the hidden dimension of the LLM (e.g., about 4096 for LLaMA-8B), and we retrieve  $t = 6$  documents for each  $q$ , then our method can roughly save  $6|d|$  tokens in inference. Compared to its offline cost, this means that Parametric RAG is more cost-friendly than in-context RAG when the number of queries is more than twice that of documents in the life cycle of the service.

In summary, while the offline preprocessing step in Parametric RAG introduces additional computational overhead compared to traditional RAG, our analysis demonstrates that, when the system handles a large number of queries, Parametric RAG can provide a more carbon-efficient solution for large-scale RAG systems.

**3.4.2 Storage Overhead.** In Parametric RAG, storage overhead comes from the Parametric Representation of each document, which consists of low-rank matrices from the FFN layer. Let  $r$  be the LoRA rank,  $n$  be the number of Transformer layers,  $h$  be the hidden size, and  $l$  the intermediate size of FFN, then the number of parameters in the Parametric Representation of a document is  $2nr(h + l)$ . For example, with the LLaMA3-8B model (32 layers, hidden size 4096, intermediate size 14336), we need to store approximately 2.36M extra parameters (with  $r = 2$  as used in our experiments). Stored at 16-bit precision, this requires around 4.72MB per document.

While the storage requirements for Parametric RAG may seem substantial compared to the raw documents, there are multiple methods to reduce its cost in practice. For example, previous studies find that the access of information in real user traffic follows a long-tail distribution [38]. Taking Google as an example, about 96.55% of Web pages receive zero traffic, and only 1.94% get one to ten visits per month [39]. Therefore, creating parametric representations for a tiny set of head documents can serve the majority of user requests, which significantly reduces the storage cost of Parametric RAG. Also, as shown in our experiments, Parametric RAG can be used with in-context RAG together for downstream tasks. Thus, it can serve as a natural boost to existing RAG methods without breaking their system pipelines.

## 4 Experimental Setup

In this section, we detail the experimental framework used to evaluate Parametric RAG. We begin with the introduction of our selected benchmark datasets (§4.1). Next, we introduce our selected baseline methods (§4.2) and implementation. Finally, we provide implementation details regarding our parameterization method, retrieval strategy, and inference settings (§4.3).

### 4.1 Benchmarks and Metrics

We evaluate our approach on diverse benchmark datasets, each designed to assess different reasoning capabilities, such as multi-hop reasoning and commonsense inference. Specifically, we select the following datasets:

- **2WikiMultihopQA (2WQA)** [13] is a dataset designed to test the model’s ability to perform multi-hop reasoning by integrating information across multiple Wikipedia passages.
- **HotpotQA (HQA)** [56] also focuses on evaluating multi-hop reasoning skills, requiring models to combine information from different contexts to address a single query.
- **PopQA (PQA)** [28] assesses factual question answering, challenging the model’s ability to recall accurate knowledge and resolve ambiguity in entity representation.
- **ComplexWebQuestions (CWQ)** [47] involves answering multi-step, web-based questions, further testing the model’s capacity to retrieve and reason over large-scale web content.

For evaluation metrics, we use the F1 score to evaluate performance on question-answering tasks, as it captures the balance between precision and recall by accounting for partially correct answers. Both **2WQA** and **HQA** categorize questions based on reasoning types, with **2WQA** divided into four categories and **HQA** into two. To comprehensively compare the performance of P-RAG and other RAG baselines across different reasoning tasks, our main experimental table (Table 1) presents the performance for each sub-task separately, using the first 300 questions from each sub-dataset. Table 1 also presents the overall performance of each RAG baseline on the two datasets in the “Total” column. Since the original datasets contain uneven distributions of question types, the “Total” column is not a simple average of the sub-dataset performances.

### 4.2 Baselines

We choose the following RAG baselines for comparison:

- **Standard RAG.** This RAG method directly appends the top retrieved documents to the LLM’s input prompt. The prompt explicitly instructs the LLM to refer to the provided documents when answering the question and also includes instructions on the output format expected from the model.
- **DA-RAG** incorporates the augmented documents and QA pairs using the Data Augmentation method introduced in § 3.2.1. This baseline aims to demonstrate that the performance improvement observed in Parametric RAG does not stem from the data augmentation phase but from the in-parameter knowledge injection.
- **FLARE** [19] is a multi-round retrieval augmentation method that triggers retrieval each time it encounters an uncertain token. When the retrieval module is triggered, the last generated sentence without the uncertain tokens is defined as the query.
- **DRAGIN** [45] is a multi-round retrieval augmentation method. It triggers retrieval when an uncertain token has semantic meaning and also has a strong influence on the following tokens. When the retrieval module is triggered, it formulates the query based on the model’s internal state and preceding context.
- **P-RAG** directly injects relevant documents into the LLM’s parameters through document parameterization, enabling efficient RAG without increasing the input context length.
- **Combine Both.** This baseline combines the in-context RAG method with P-RAG, leveraging both in-context and parametric knowledge injection. This baseline aims to evaluate whether the fusion of these approaches leads to better performance.

For P-RAG and all the baselines, we use the same retriever and select the top 3 retrieved documents as relevant. To ensure a fair

**Table 1: The overall experiment results of Parametric RAG and other baselines across four tasks. All metrics reported are F1 scores. Bold numbers indicate the best performance of all baselines, and the second-best results are underlined. “\*” and † denote significantly worse performance than the bolded method and our proposed P-RAG with  $p < 0.05$  level, respectively.**

		2WikiMultihopQA					HotpotQA			PopQA	CWQ
		Compare	Bridge	Inf.	Compose	Total	Bridge	Compare	Total		
LLaMA-1B	Standard RAG	0.4298 <sup>†*</sup>	0.3032 <sup>†*</sup>	0.2263	0.1064	0.2520 <sup>*</sup>	0.2110	0.4083	0.2671	0.1839 <sup>*</sup>	0.3726
	DA-RAG	0.3594 <sup>†*</sup>	0.2587 <sup>†*</sup>	0.2266	0.0869 <sup>†*</sup>	0.2531 <sup>*</sup>	0.1716 <sup>*</sup>	0.3713 <sup>†*</sup>	0.2221	0.2012 <sup>*</sup>	0.3691
	FLARE	0.4013 <sup>†*</sup>	0.2589 <sup>†*</sup>	0.1960	0.0823 <sup>†*</sup>	0.2234 <sup>*</sup>	0.1630 <sup>*</sup>	0.3784 <sup>†*</sup>	0.1785 <sup>*</sup>	0.1301 <sup>†*</sup>	0.3173 <sup>*</sup>
	DRAGIN	0.4556	0.3357 <sup>*</sup>	0.1919	0.0901	0.2692 <sup>*</sup>	0.1431 <sup>*</sup>	0.4015	0.1830 <sup>*</sup>	0.1056 <sup>†*</sup>	0.3900
	P-RAG (Ours)	0.4920	0.3994	0.2185	0.1334	0.2764	0.1602 <sup>*</sup>	<b>0.4493</b>	0.1999 <sup>*</sup>	0.2205 <sup>*</sup>	0.3482 <sup>*</sup>
	Combine Both	<b>0.5046</b>	<b>0.4595</b>	<b>0.2399</b>	<b>0.1357</b>	<b>0.3237</b>	<b>0.2282</b>	0.4217	<b>0.2689</b>	<b>0.2961</b>	<b>0.4101</b>
Qwen-1.5B	Standard RAG	0.3875 <sup>†*</sup>	0.3884 <sup>†*</sup>	0.1187 <sup>†*</sup>	0.0568 <sup>†*</sup>	0.2431 <sup>†*</sup>	0.1619 <sup>*</sup>	0.3713 <sup>†*</sup>	0.2073 <sup>*</sup>	0.0999 <sup>†*</sup>	0.2823 <sup>*</sup>
	DA-RAG	0.3418 <sup>†*</sup>	0.4015	0.1269 <sup>†*</sup>	0.0514 <sup>†*</sup>	0.2156 <sup>†*</sup>	0.1182 <sup>†*</sup>	0.3041 <sup>†*</sup>	0.1683 <sup>*</sup>	0.1197 <sup>†*</sup>	0.2718 <sup>†*</sup>
	FLARE	0.1896 <sup>†*</sup>	0.1282 <sup>†*</sup>	0.0852 <sup>†*</sup>	0.0437 <sup>†*</sup>	0.1004 <sup>†*</sup>	0.0750 <sup>†*</sup>	0.1229 <sup>†*</sup>	0.0698 <sup>†*</sup>	0.0641 <sup>†*</sup>	0.1647 <sup>†*</sup>
	DRAGIN	0.2771 <sup>†*</sup>	0.1826 <sup>†*</sup>	0.1025 <sup>†*</sup>	0.0680 <sup>†*</sup>	0.1538 <sup>†*</sup>	0.0801 <sup>†*</sup>	0.1851 <sup>†*</sup>	0.0973 <sup>†*</sup>	0.0548 <sup>†*</sup>	0.1788 <sup>†*</sup>
	P-RAG (Ours)	<b>0.4529</b>	<b>0.4494</b>	<b>0.2072</b>	<b>0.1372</b>	<b>0.3025</b>	0.1720	0.4623	0.2165 <sup>*</sup>	0.1885	0.3280
	Combine Both	0.4053	0.4420	0.1705	0.1154	0.2627	<b>0.2383</b>	<b>0.5037</b>	<b>0.2942</b>	<b>0.2261</b>	<b>0.3495</b>
LLaMA-8B	Standard RAG	0.5843 <sup>†*</sup>	0.4794 <sup>†*</sup>	0.1833 <sup>†*</sup>	0.0991 <sup>†*</sup>	0.3372 <sup>†*</sup>	0.1823 <sup>†*</sup>	0.3493 <sup>†*</sup>	0.2277 <sup>†*</sup>	0.1613 <sup>†*</sup>	0.3545 <sup>†*</sup>
	DA-RAG	0.4921 <sup>†*</sup>	0.3344 <sup>†*</sup>	0.1523 <sup>†*</sup>	0.0670 <sup>†*</sup>	0.2396 <sup>†*</sup>	0.1587 <sup>†*</sup>	0.2860 <sup>†*</sup>	0.1996 <sup>†*</sup>	0.2255 <sup>*</sup>	0.3481 <sup>†*</sup>
	FLARE	0.4293 <sup>†*</sup>	0.3769 <sup>†*</sup>	0.3086	0.1627 <sup>*</sup>	0.3492 <sup>*</sup>	0.2493 <sup>†*</sup>	0.4324 <sup>†*</sup>	0.2771 <sup>†*</sup>	0.2393 <sup>*</sup>	0.3084 <sup>†*</sup>
	DRAGIN	0.5185 <sup>†*</sup>	0.4480 <sup>†*</sup>	0.2664	0.1833	0.3544 <sup>*</sup>	0.2618 <sup>*</sup>	0.6116 <sup>*</sup>	0.2924 <sup>*</sup>	0.1772 <sup>†*</sup>	0.3101 <sup>†*</sup>
	P-RAG (Ours)	0.6353	0.5437	0.2471 <sup>*</sup>	0.1992	0.3932	0.3115 <sup>*</sup>	0.6557	0.3563 <sup>*</sup>	0.2413 <sup>*</sup>	0.4541
	Combine Both	<b>0.6432</b>	<b>0.5556</b>	<b>0.3160</b>	<b>0.2339</b>	<b>0.4258</b>	<b>0.4025</b>	<b>0.6918</b>	<b>0.4559</b>	<b>0.3059</b>	<b>0.4728</b>

comparison, we ensured that P-RAG and all the baselines share the same prompt template<sup>6</sup> under the same dataset.

### 4.3 Implementation Details

In this subsection, we introduce the specific implementation of our experiments:

**Base Models** We implement Parametric RAG using open-source pre-trained LLMs. To ensure the broad effectiveness of P-RAG across different models, we selected LLMs of varying scales and from different series, including Qwen2.5-1.5B-Instruct [55], LLaMA-3.2-1B-Instruct [29], and Llama-3-8B-Instruct [30]. All experiments were conducted using PyTorch on NVIDIA A100 GPUs with 40GB of memory.

**Preprocessing and Parameterization.** Consistent with prior works [19, 20, 44, 45], we utilize Wikipedia dumps as our external knowledge corpus, specifically adopting the dataset<sup>7</sup> proposed by DPR [20]. For document augmentation, each document is rewritten once, and three QA pairs are generated based on the document<sup>8</sup> (using the downstream LLM, if not mentioned explicitly). In the LoRA fine-tuning process, the learning rate was set to  $3 \times 10^{-4}$ , and the training epoch was set to 1. The LoRA modules were exclusively integrated into the feed-forward network (FFN) matrices, excluding the query, key, and value ( $QKV$ ) matrices. The scaling factor  $\alpha$  was configured to 32, LoRA rank  $r$  was set to 2, and no dropout was applied during training to ensure stability and full utilization of

the parameter updates. The LoRA weight is randomly initialized following the setting of the original LoRA paper [14].

**Retrieval Module.** Recent studies on retrieval-augmented generation (RAG) [33] reveal that BM25 performs on par with, or even outperforms, state-of-the-art dense models in some scenarios. Given its strong performance, simplicity, and low computational cost, we adopt BM25 as the retriever for our approach. We use Elasticsearch as the backend for implementing BM25, with detailed configuration settings and instructions available on our official GitHub repository.

**Generation Configuration.** All experiments are conducted using the publicly released Hugging Face implementations of LLaMA and Qwen. We adopt the default hyperparameters and chat template provided in the official Huggingface repository, with the only modification being the use of greedy decoding to ensure the reproducibility of our reported results.

## 5 Experiments

### 5.1 Main Experiment

In this section, we present the main experimental result and an in-depth analysis of our proposed Parametric RAG compared with other RAG baselines and a combined setting that leverages both parametric and in-context knowledge injection. The experimental results are presented in Table 1, and we provide the following analysis: **(1) Overall Analysis.** P-RAG outperforms existing RAG frameworks in most of the benchmarks and LLMs evaluated. This trend is especially obvious for Qwen-1.5B and LLaMA-8B. The improvements suggest that the incorporation of knowledge into model parameters can enhance the overall performance of the RAG pipeline, enabling the model to recall and reason over the injected

<sup>6</sup>All the prompt templates used in this paper are available in our GitHub repository: [https://github.com/oneal2000/PRAG/blob/main/all\\_prompt.md](https://github.com/oneal2000/PRAG/blob/main/all_prompt.md)

<sup>7</sup><https://github.com/facebookresearch/DPR/tree/main>

<sup>8</sup>The detailed prompt template for document augmentation is publicly available on our official GitHub repository.

**Table 2: Ablation study on the impact of LoRA weight initialization strategies for P-RAG. All metrics reported are F1 scores. “P-RAG Rand.” and “P-RAG Warm.” indicate randomly initialized LoRA weights and warm-up LoRA initialization, respectively. The best results are in bold.**

		2WQA	HQA	PQA	CWQ
LLaMA-1B	P-RAG Rand.	0.2764	0.1999	<b>0.2205</b>	0.3482
	P-RAG Warm.	<b>0.3546</b>	<b>0.2456</b>	0.2035	<b>0.4263</b>
Qwen-1.5B	P-RAG Rand.	0.3025	0.2165	0.1885	0.3280
	P-RAG Warm.	<b>0.3542</b>	<b>0.2718</b>	<b>0.2418</b>	<b>0.5018</b>
LLaMA-8B	P-RAG Rand.	0.3932	0.3563	0.2413	0.4541
	P-RAG Warm.	<b>0.4201</b>	<b>0.4499</b>	<b>0.2952</b>	<b>0.5591</b>

knowledge more effectively. Furthermore, since these gains are observed in models from different series and parameter sizes, the results underscore the robustness and broad applicability of P-RAG. (2) **Comparison with DA-RAG.** DA-RAG incorporates all the content generated during the Document Augmentation phase into the context, whereas our proposed P-RAG consistently outperforms DA-RAG across all settings. This result demonstrates that the performance improvement observed in Parametric RAG does not stem from the document augmentation phase, but from the in-parameter knowledge injection paradigm. (3) **Impact of Model Scale on P-RAG.** The performance gap between P-RAG and other RAG baselines is noticeably more significant when moving from the LLaMA-1B model to LLaMA-8B. This discrepancy indicates that parametric injection becomes even more beneficial in larger-scale models because larger models can better leverage internalized document knowledge. (4) **Combine In-context RAG and P-RAG.** The combined use of parametric and in-context RAG methods (Combine Both) yields the highest overall performance across various datasets and base LLMs. This result highlights that in-parameter knowledge injection is not in conflict with traditional RAG methods based on in-context knowledge injection. Consequently, our proposed document parameterization approaches can be seamlessly integrated for downstream tasks, allowing Parametric RAG to enhance existing RAG systems without disrupting their pipelines. (5) **Other Findings.** Both DRAGIN and FLARE underperform significantly when applied to Qwen-2.5-1.5B. Our analysis suggests that Qwen-2.5-1.5B tends to produce highly confident answers regardless of uncertainty. Since these dynamic RAG frameworks rely on confidence to trigger retrieval, they rarely activate on Qwen-2.5-1.5B. This highlights a key limitation of uncertainty-based triggers and underscores the need for more robust mechanisms in dynamic RAG frameworks.

## 5.2 Impact of LoRA Weight Initialization

To investigate the impact of LoRA weight initialization strategies on our proposed Parametric RAG framework, we conducted an ablation study using two initialization strategies:

(1) **Random Initialization** (P-RAG Rand.): The LoRA weights are initialized randomly without any pretraining or warm-up, which represents the default setting for our proposed Parametric RAG

approach. All the Parametric RAG experimental results reported in other sections in this paper are based on this setting.

(2) **Warm-Up Initialization** (P-RAG Warm.): LoRA weights are pre-trained using a set of 600 sampled question-answer (QA) pairs. These QA pairs are selected from the training sets of our chosen benchmarks and are distinct from the test questions to ensure no data leakage. The pre-training process involves training the LoRA parameters using the standard next-token prediction method on the concatenated tokens of the QA pairs. The specific implementation follows Section 4.3, and the pre-trained LoRA parameters are saved as initialization for the Document Parameterization phase<sup>9</sup>.

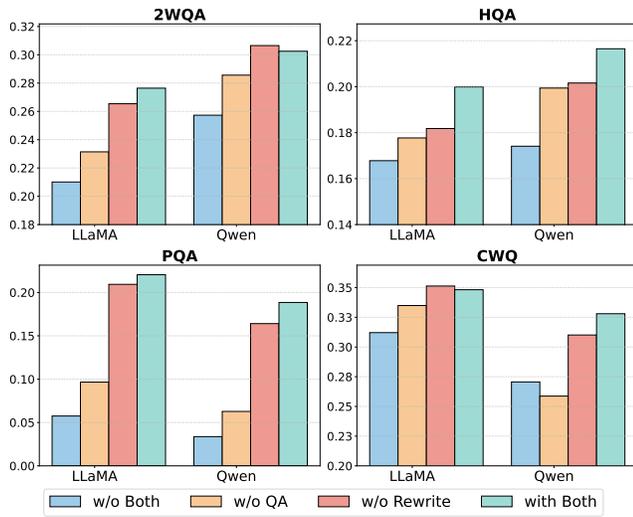
The experimental results in Table 2 clearly indicate that across different model series and scales, as well as diverse datasets, the warm-up initialization strategy (P-RAG Warm.) consistently outperforms random initialization (P-RAG Rand.). This demonstrates the effectiveness of task-aware pretraining in enhancing the Parametric RAG pipeline. Furthermore, the observed improvements across varying model sizes confirm the scalability and generality of this approach. The superior performance of the warm-up approach in downstream tasks can be attributed to two key factors. First, it effectively aligns the additional LoRA parameters with the base LLM before document parameterizing, ensuring a smoother integration of knowledge. Second, it facilitates the incorporation of task-relevant knowledge, including output formats and generation patterns, which are critical for enhancing the quality of response in certain tasks. This finding suggests that in practical Parametric RAG applications where the downstream task is fixed, warming up the LoRA parameters for the task offers a promising approach to boost effectiveness. It is important to note that our main experiments (as well as all other experiments in this paper) were conducted using random initialization without any task-specific optimizations or dataset-specific tuning. This further highlights the strong generalization capability of our proposed Parametric RAG paradigm.

These findings also highlight a broader insight: embedding few-shot examples either in the model’s context or directly into its parameters leads to improved downstream task performance. Interestingly, our proposed parametric information representation method offers compatibility with few-shot in-context learning, enabling a combination of parametric and in-context knowledge augmentation.

## 5.3 Impact of Document Augmentation

To investigate the individual contributions of the rewriting and question-answer (QA) generation steps in the document augmentation process, we conduct a series of ablation experiments by removing (1) both rewriting and QA, (2) QA alone, and (3) rewriting alone. The experimental results are shown in Figure 3, and we have the following observations: (1) When neither rewriting nor QA generation is employed, the performance consistently degrades significantly across all evaluated tasks and models. This reduction suggests that simply training the LLM on the selected document via the next token prediction task without any form of data augmentation leads to insufficient internalization of facts by the model. (2)

<sup>9</sup>All the training code and data are publicly available at our anonymous GitHub repository: <https://github.com/oneal2000/PRAG>



**Figure 3: Ablation study on the impact of the document augmentation stage. LLaMA indicates LLaMA-3.2-1B, and Qwen indicates Qwen-2.5-1.5B. The metric used is the F1 Score.**

Removing either QA or rewriting alone yields better results than removing both, indicating that each step offers distinct benefits. However, we notice that removing QA leads to a more significant performance decline than removing rewriting. This observation suggests that QA pair generation is more crucial for pushing the model to recall and apply factual information while rewriting offers valuable diversity in phrasing and structure and benefits the overall performance. (3) Incorporating both rewriting and QA results in the strongest overall performance on most of the evaluated tasks and models. These findings reinforce that rewriting and QA generation play complementary roles.

In general, this ablation study indicates that both rewriting and QA generation significantly enhance the performance of document parameterizing. Their integration produces the best performance. Rewriting expands the coverage and diversity of context, while QA explicitly encourages the model to encode the knowledge of the selected document in a necessary way to apply the knowledge for downstream tasks. Therefore, it is advisable to incorporate both components of our document augmentation for effective internalization of knowledge.

#### 5.4 Impact of Data-augmentation Model

To evaluate the impact of the choice of LLM in the Document Augmentation phase, we conducted an ablation study comparing different configurations of the model used for document rewriting and QA pair generation. In our default setting, we use the same LLM for both the document augmentation process and the downstream task. However, to explore whether the performance of our method is sensitive to this choice, we tested alternative configurations. Specifically, we tested different model sizes by using both smaller and larger LLMs for document augmentation and QA pair generation. The experimental results are shown in Table 3, indicating that our framework demonstrates an insensitivity to the

**Table 3: Ablation study comparing different document augmentation models. GenLM indicates the generator LLM and AugLM indicates the LLM for document augmentation. LLaMA indicates LLaMA-3.2-1B, and Qwen indicates Qwen-2.5-1.5B. The best results are in bold. The metric used in the table is F1 Score.**

GenLM	AugLM	Dataset			
		2WQA	HQA	PQA	CWQ
LLaMA-1B	LLaMA-1B	0.2764	0.1999	0.2205	0.3482
	Qwen-1.5B	0.2753	0.1980	0.2340	0.3495
	LLaMA-8B	0.2748	0.1935	0.2207	0.3498
Qwen-1.5B	LLaMA-1B	0.2974	0.2005	0.1829	0.3183
	Qwen-1.5B	0.3025	0.2165	0.1885	0.3280
	LLaMA-8B	0.2948	0.2161	0.2156	0.3211

choice of data augmentation model. Performance remains consistent across different setups, regardless of whether a smaller, larger, or the same model as the generator is used for data augmentation. Importantly, using a small model for augmentation yields comparable results to employing significantly larger models, indicating that the augmentation step does not require high-capacity models to be effective. Similarly, when the same model is used for both generation and augmentation, the outcomes are indistinguishable from those where separate models are employed.

#### 5.5 Runtime Analysis

We present the inference time for the LLaMA3-8B model across various RAG baselines on 2WikiMultihopQA (2WQA) and ComplexWebQuestions (CWQ) in Table 4. This evaluation simulates the online inference latency for answering a question using different RAG approaches. All experiments were conducted on the same GPU server to ensure consistent evaluation conditions. The experimental results indicate that P-RAG reduces the time per question by 29% to 36% compared to Standard RAG. Notably, the Combine Both baseline, which showed the best performance and significant improvements in the main experiment, requires almost the same online computation time as the Standard RAG method. In contrast, multi-round RAG frameworks like DRAGIN and FLARE exhibit significantly higher latency for answering a question compared to single-round methods. For both P-RAG and Combine Both baselines, we present the inference times separately from the time required for merging and loading the LoRA (0.32s). This distinction arises because, in our current implementation, the time spent on merging and loading the LoRA significantly exceeds theoretical expectations. The floating-point operations involved in the LoRA operation step contribute less than 1% to the total computational cost of generating a response [14], but the latency of memory loading and data communications in our current implementation is far from perfect. We believe this latency can potentially be addressed through engineering optimizations. It is important to note that our analysis primarily emphasizes the relative time ratios and trends across the different methods, as actual application times and latencies can vary depending on hardware configurations, such as CPU, GPU, memory, and storage.

**Table 4: The average time required by the LLaMA3-8B model to answer a question on the 2WikiMultihopQA (2WQA) and ComplexWebQuestions (CWQ) datasets. The "+0.32" footnote for P-RAG and Combine Both indicates the total time needed for merging and loading the LoRA adapter.**

	2WQA		CWQ	
	Time(s)	Speed Up	Time(s)	Speed Up
<b>P-RAG</b>	2.34 <sub>+0.32</sub>	1.29x	2.07 <sub>+0.32</sub>	1.36x
<b>Combine Both</b>	3.08 <sub>+0.32</sub>	0.98x	2.84 <sub>+0.32</sub>	0.99x
<b>Standard RAG</b>	3.03	1.00x	2.82	1.00x
<b>FLARE</b>	10.14	0.25x	11.31	0.25x
<b>DRAGIN</b>	14.60	0.21x	16.21	0.17x

## 6 Conclusion and Future Directions

This work introduces Parametric RAG, a novel framework that addresses the limitations of in-context knowledge augmentation by parameterizing external documents. Parametric RAG infuses these parameterized documents directly into the model, reducing contextual overload and online computational costs while maintaining robust performance. Our experiments on multiple benchmarks demonstrate that Parametric RAG outperforms traditional retrieval-augmented generation methods across different LLMs. Ultimately, Parametric RAG offers a more efficient and scalable pathway to integrate external knowledge into LLMs, paving the way for further innovation in parametric-based knowledge augmentation.

Despite its significant potential, Parametric RAG presents several challenges that warrant further investigation. First, the current parameterization process is computationally intensive, and the parametric representations of each document are substantially larger than plain text. Future work could explore more methods to improve computational and storage efficiency, making the parameterization process more scalable. Second, the parameterized documents are currently tied to specific LLMs, restricting their ability to generalize across different models. Developing universal, model-agnostic representations could significantly enhance flexibility and reuse across diverse systems. Finally, we believe the potential applications of information parameterization can be extended beyond RAG. For instance, LLM-based agents could benefit from parameterizing the agent’s profiles and configuration, which could alleviate context-length constraints and improve online computational efficiency. By addressing these challenges, future research could unlock more potential for the Parametric RAG paradigm.

## References

- [1] Zeyuan Allen-Zhu and Yuanzhi Li. [n. d.]. Physics of Language Models: Part 3.1, Knowledge Storage and Extraction. In *Forty-first International Conference on Machine Learning*.
- [2] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. [n. d.]. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In *The Twelfth International Conference on Learning Representations*.
- [3] Ingeol Baek, Hwan Chang, Byeongjeong Kim, Jimin Lee, and Hwanhee Lee. 2024. Probing-RAG: Self-Probing to Guide Language Models in Selective Document Retrieval. *arXiv preprint arXiv:2410.13339* (2024).
- [4] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*. PMLR, 2206–2240.
- [5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.
- [6] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311* (2022).
- [7] Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. 2023. How abilities in large language models are affected by supervised fine-tuning data composition. *arXiv preprint arXiv:2310.05492* (2023).
- [8] Qian Dong, Qingyao Ai, Hongning Wang, Yiding Liu, Haitao Li, Weihang Su, Yiqun Liu, Tat-Seng Chua, and Shaoping Ma. 2025. Decoupling Knowledge and Context: An Efficient and Effective Retrieval Augmented Generation Framework via Cross Attention. In *Proceedings of the ACM on Web Conference 2025*.
- [9] Qian Dong, Yiding Liu, Qingyao Ai, Haitao Li, Shuaiqiang Wang, Yiqun Liu, Dawei Yin, and Shaoping Ma. 2023. I3 retriever: incorporating implicit interaction in pre-trained language models for passage retrieval. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 441–451.
- [10] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130* (2024).
- [11] Yan Fang, Jingtao Zhan, Qingyao Ai, Jiaxin Mao, Weihang Su, Jia Chen, and Yiqun Liu. 2024. Scaling laws for dense retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1339–1349.
- [12] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*. PMLR, 3929–3938.
- [13] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060* (2020).
- [14] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.
- [15] Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. 2024. GRAG: Graph Retrieval-Augmented Generation. *arXiv preprint arXiv:2405.16506* (2024).
- [16] Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282* (2020).
- [17] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, Kevin Duh, Helena Gomez, and Steven Bethard (Eds.). Association for Computational Linguistics, Mexico City, Mexico, 7036–7050. <https://doi.org/10.18653/v1/2024.naacl-long.389>
- [18] Zhengbao Jiang, Luyu Gao, Jun Araki, Haibo Ding, Zhiruo Wang, Jamie Callan, and Graham Neubig. 2022. Retrieval as attention: End-to-end learning of retrieval and reading within a single transformer. *arXiv preprint arXiv:2212.02027* (2022).
- [19] Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983* (2023).
- [20] 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. *arXiv preprint arXiv:2004.04906* (2020).
- [21] Zixuan Ke, Weize Kong, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 2024. Bridging the preference gap between retrievers and llms. *arXiv preprint arXiv:2401.06954* (2024).
- [22] Mosh Levy, Alon Jacoby, and Yoav Goldberg. 2024. Same task, more tokens: the impact of input length on the reasoning performance of large language models. *arXiv preprint arXiv:2402.14848* (2024).
- [23] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, Mike Lewis, Wen-tau Yih, Tim Rocktaschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.
- [24] Haitao Li, Jia Chen, Weihang Su, Qingyao Ai, and Yiqun Liu. 2023. Towards better web search performance: pre-training, fine-tuning and learning to rank. *arXiv preprint arXiv:2303.04710* (2023).
- [25] Huanshuo Liu, Hao Zhang, Zhijiang Guo, Kuicai Dong, Xiangyang Li, Yi Quan Lee, Cong Zhang, and Yong Liu. 2024. CtrlA: Adaptive Retrieval-Augmented Generation via Probe-Guided Control. *arXiv preprint arXiv:2405.18727* (2024).
- [26] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models

- use long contexts. *Transactions of the Association for Computational Linguistics* 12 (2024), 157–173.
- [27] Yixiao Ma, Yueyue Wu, Weihang Su, Qingyao Ai, and Yiqun Liu. 2023. CaseEncoder: A Knowledge-enhanced Pre-trained Model for Legal Case Encoding. *arXiv preprint arXiv:2305.05393* (2023).
- [28] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khoshabi, and Hannaneh Hajishirzi. 2023. When Not to Trust Language Models: Investigating Effectiveness of Parametric and Non-Parametric Memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 9802–9822. <https://doi.org/10.18653/v1/2023.acl-long.546>
- [29] Meta. 2024. Llama-3.2-1B-Instruct. <https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct> Accessed: 2024-09.
- [30] Meta. 2024. Meta-Llama-3-8B-Instruct. <https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct> Accessed: 2024-04.
- [31] Neel Nanda, Senthoooran Rajamanoharan, János Kramár, and Rohin Shah. 2023. *Fact Finding: Attempting to Reverse-Engineer Factual Recall on the Neuron Level*. <https://www.lesswrong.com/posts/iGuwZTHWb6DFY3sKB/fact-finding-attempting-to-reverse-engineer-factual-recall> Accessed: 2025-01-24.
- [32] Boci Peng, Yun Zhu, Yongchao Liu, Xiaohe Bo, Haizhou Shi, Chuntao Hong, Yan Zhang, and Siliang Tang. 2024. Graph retrieval-augmented generation: A survey. *arXiv preprint arXiv:2408.08921* (2024).
- [33] Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *arXiv preprint arXiv:2302.00083* (2023).
- [34] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.
- [35] Alireza Salemi and Hamed Zamani. 2024. Towards a search engine for machines: Unified ranking for multiple retrieval-augmented large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 741–751.
- [36] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100* (2022).
- [37] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652* (2023).
- [38] Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. 1999. Analysis of a very large web search engine query log. *SIGIR Forum* 33, 1 (Sept. 1999), 6–12. <https://doi.org/10.1145/331403.331405>
- [39] Tim Soulo. 2023. *96.55% of Content Gets No Traffic From Google. Here's How to Be in the Other 3.45% [New Research for 2023]*. <https://ahrefs.com/blog/search-traffic-study/> Accessed: 2025-01-24.
- [40] Weihang Su, Qingyao Ai, Xiangsheng Li, Jia Chen, Yiqun Liu, Xiaolong Wu, and Shengluan Hou. 2023. Wikiformer: Pre-training with Structured Information of Wikipedia for Ad-hoc Retrieval. *arXiv preprint arXiv:2312.10661* (2023).
- [41] Weihang Su, Qingyao Ai, Yueyue Wu, Yixiao Ma, Haitao Li, and Yiqun Liu. 2023. Caseformer: Pre-training for Legal Case Retrieval. *arXiv preprint arXiv:2311.00333* (2023).
- [42] Weihang Su, Yiran Hu, Anzhe Xie, Qingyao Ai, Quezi Bing, Ning Zheng, Yun Liu, Weixing Shen, and Yiqun Liu. 2024. STARD: A Chinese Statute Retrieval Dataset Derived from Real-life Queries by Non-professionals. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 10658–10671. <https://doi.org/10.18653/v1/2024.findings-emnlp.625>
- [43] Weihang Su, Xiangsheng Li, Yiqun Liu, Min Zhang, and Shaoping Ma. 2023. Thuir2 at ntcir-16 session search (ss) task. *arXiv preprint arXiv:2307.00250* (2023).
- [44] Weihang Su, Yichen Tang, Qingyao Ai, Changyue Wang, Zhijing Wu, and Yiqun Liu. 2024. Mitigating entity-level hallucination in large language models. In *Proceedings of the 2024 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*. 23–31.
- [45] Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. 2024. DRAGIN: Dynamic Retrieval Augmented Generation based on the Real-time Information Needs of Large Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 12991–13013. <https://doi.org/10.18653/v1/2024.acl-long.702>
- [46] Weihang Su, Changyue Wang, Qingyao Ai, Yiran Hu, Zhijing Wu, Yujia Zhou, and Yiqun Liu. 2024. Unsupervised real-time hallucination detection based on the internal states of large language models. *arXiv preprint arXiv:2403.06448* (2024).
- [47] Alon Talmor and Jonathan Berant. 2018. The Web as a Knowledge-Base for Answering Complex Questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, Marilyn Walker, Heng Ji, and Amanda Stent (Eds.). Association for Computational Linguistics, New Orleans, Louisiana, 641–651. <https://doi.org/10.18653/v1/N18-1059>
- [48] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).
- [49] Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509* (2022).
- [50] Changyue Wang, Weihang Su, Qingyao Ai, and Yiqun Liu. 2024. Knowledge Editing through Chain-of-Thought. *arXiv preprint arXiv:2412.17727* (2024).
- [51] Changyue Wang, Weihang Su, Hu Yiran, Qingyao Ai, Yueyue Wu, Cheng Luo, Yiqun Liu, Min Zhang, and Shaoping Ma. 2024. LeKUBE: A Legal Knowledge Update BEnchmark. *arXiv preprint arXiv:2407.14192* (2024).
- [52] Yile Wang, Peng Li, Maosong Sun, and Yang Liu. 2023. Self-knowledge guided retrieval augmentation for large language models. *arXiv preprint arXiv:2310.05002* (2023).
- [53] Zihao Wang, Anji Liu, Haowei Lin, Jiaqi Li, Xiaojian Ma, and Yitao Liang. 2024. Rat: Retrieval augmented thoughts elicit context-aware reasoning in long-horizon generation. *arXiv preprint arXiv:2403.05313* (2024).
- [54] Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. 2024. Continual learning for large language models: A survey. *arXiv preprint arXiv:2402.01364* (2024).
- [55] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 Technical Report. *arXiv preprint arXiv:2412.15115* (2024).
- [56] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600* (2018).
- [57] Zijun Yao, Weijian Qi, Liangming Pan, Shulin Cao, Linmei Hu, Weichuan Liu, Lei Hou, and Juanzi Li. 2024. Seekr: Self-aware knowledge retrieval for adaptive retrieval augmented generation. *arXiv preprint arXiv:2406.19215* (2024).
- [58] Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Rankrag: Unifying context ranking with retrieval-augmented generation in llms. *arXiv preprint arXiv:2407.02485* (2024).
- [59] Zeping Yu and Sophia Ananiadou. 2024. Neuron-Level Knowledge Attribution in Large Language Models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 3267–3280. <https://doi.org/10.18653/v1/2024.emnlp-main.191>
- [60] ChengXiang Zhai. 2008. Statistical language models for information retrieval. *Synthesis lectures on human language technologies* 1, 1 (2008), 1–141.