Improving End-to-End Training of Retrieval-Augmented Generation Models via Joint Stochastic Approximation

Hongyu Cao[1] *Yuxuan Wu*[1] *Yucheng Cai*[1] *Xianyu Zhao*[3] *Zhijian Ou*[1,2,3*]

[1]SPMI Lab, Tsinghua University, China; [2]IEIT in Tianjin, Tsinghua University, China
[3]TasiTech Co., Ltd., China

ozj@tsinghua.edu.cn

Abstract

Retrieval-augmented generation (RAG) has become a widely recognized paradigm to combine parametric memory with non-parametric memory. An RAG model consists of two serial connecting components (retriever and generator). A major challenge in end-to-end optimization of the RAG model is that marginalization over relevant passages (modeled as discrete latent variables) from a knowledge base is required. Traditional top-K marginalization and variational RAG (VRAG) suffer from biased or high-variance gradient estimates. In this paper, we propose and develop joint stochastic approximation (JSA) based end-to-end training of RAG, which is referred to as JSA-RAG. The JSA algorithm is a stochastic extension of the EM (expectation-maximization) algorithm and is particularly powerful in estimating discrete latent variable models. Extensive experiments are conducted on five datasets for two tasks (open-domain question answering, knowledge-grounded dialogs) and show that JSA-RAG significantly outperforms both vanilla RAG and VRAG. Further analysis shows the efficacy of JSA-RAG from the perspectives of generation, retrieval, and low-variance gradient estimate.

1. Introduction

Large language models (LLMs) have been shown to store factual knowledge in their parameters through pre-training over large amounts of Internet corpora [1,2]. However, such implicit knowledge cannot be easily updated, expanded, inspected, and interpreted. Moreover, for many knowledge-intensive tasks, the use of external knowledge beyond the parametric memory of LLMs to generate responses is critical, such as in open-domain question answering (ODQA) [3–5] and knowledge-grouned dialog systems [6–8]. To address these issues, hybrid models that combine parametric memory with nonparametric memories have emerged [4, 9], among which retrieval-augmented generation (RAG) has drawn considerable attention [10].

During recent years, RAG has not only been used to refer to the particular method developed in [10], but also, more often, represents a general two-step paradigm (retrieve-then-generate). In the RAG paradigm, given a context (denoted by x) such as a query in QA or a dialog context, relevant passages (denoted by h) are first obtained from external knowledge bases (KBs) by using a retriever. The retrieved passages are then combined with the context and fed into a generator to generate the response y.

Hence, a RAG model consists of two serial connecting components (retriever and generator). During training, if the relevant passage is known (e.g., human-annotated gold passage), we can supervise the retriever with that passage, and train the generator conditioned on that passage as well. However, collecting human annotations of gold passages is labor intensive. This challenge leads to the widely adopted approach of training retrievers and generators separately. Retrievers are often trained on one corpus [5, 11, 12], and then generators are trained on another different corpus using fixed retrievers [13–15]. While this is fairly easy to implement, separate training is sub-optimal, for example, the retriever never improves as the generator learns to generate responses.

There have been efforts in developing end-to-end training of an RAG model [10, 16–18], which means eliminating the reliance on intermediate annotations and training all model components simultaneously. In [10], to train the retriever and generator end-to-end, the relevant passage is treated as a discrete latent variable and the following marginal log-likelihood is to be maximized:

$$p(y|x) = \sum_h p(h|x)p(y|x,h) \tag{1}$$

Thus end-to-end training of RAG in essence amounts to unsupervised training of a discrete latent-variable model, as shown above. Direct marginalization is intractable; hence, originally, top-K marginalization (TKM) is used for the approximation [10], which we refer to as vanilla RAG. Recently, variational learning (VL) [19] has been applied to end-to-end training of RAG in two concurrent and similar works - VRAG [7] and Hindsight [20], which we refer to collectively as VRAG. In VRAG, an auxiliary inference model is introduced, acting as a posterior retriever. However, for variational learning of discrete latent variable models, the traditional Monte Carlo gradient estimator for the inference model parameter is known to be either biased or have high-variance [21].

Recently, the joint stochastic approximation (JSA) algorithm [21,22] has emerged to learn discrete latent variable models with better performance than VL. JSA is a stochastic extension of the EM (expectation-maximization) algorithm and gives unbiased, low-variance stochastic gradients for the inference model.

In this paper, we propose JSA based end-to-end training of RAG, which is referred to as JSA-RAG, as overviewed in Figure 1. JSA-RAG makes the following contributions. First, we design all model components (including prior retriever, generator, and posterior retriever) and implement the whole training and decoding pipeline to enable the successful application of JSA. We address some computational challenges to work with large-scale KBs (e.g., tens of millions of passages in Wikipedia). Second, we investigate the effect of index rebuilding in training. We study the passage concatenation strategy for post-training of

*Corresponding author. This work is supported by National Science and Technology Major Project (2023ZD0121401), Guangxi Science and Technology Project (GUIKEAB25069456).

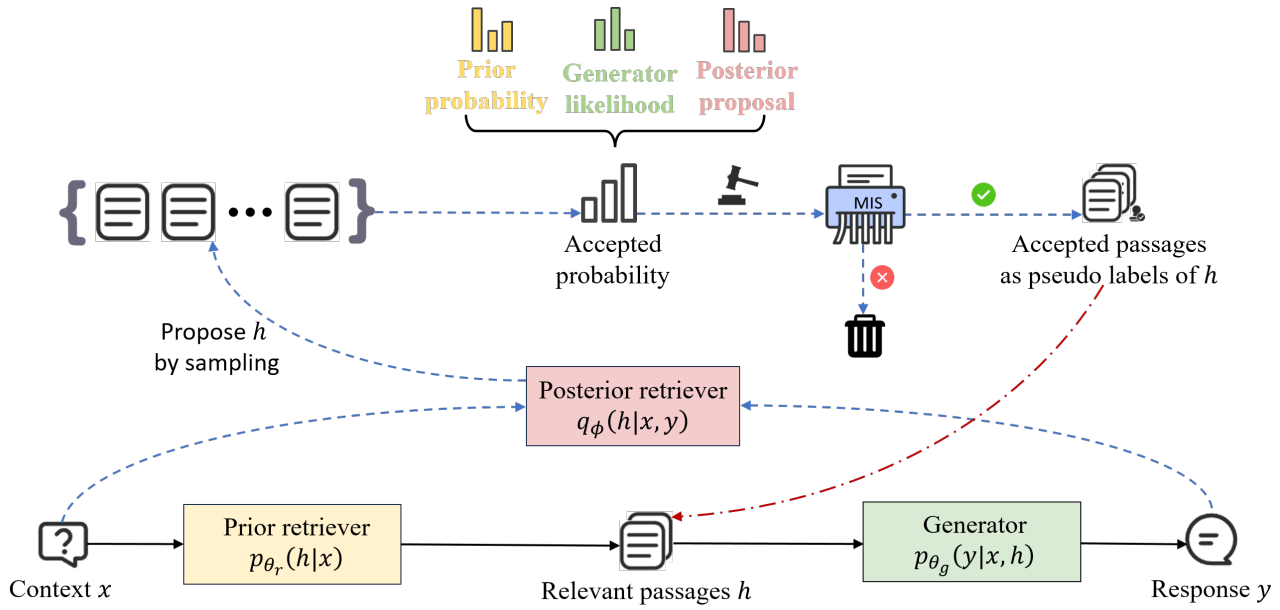

Figure 1: *Overview of JSA-RAG. 1) In addition to the (prior) retriever and generator, JSA-RAG introduces an (auxiliary) posterior retriever. 2) During training, the posterior retriever proposes relevant passages, which get accepted or rejected according to the probabilities calculated from the three components. The blue dashed line shows such Metropolis independence sampling (MIS), which is a Monte Carlo approximation of the E-step in EM. 3) The filtered passages are then treated as pseudo labels, as shown by the red dotted line. 4) Given the pseudo labels, we can calculate the gradients for prior retriever, posterior retriever, and generator, respectively, and proceed with parameter updating, very similar to perform supervised training, like the M-step in EM.*

the generator while fixing the retriever. These further demonstrate the capability and bonus offered by JSA-RAG. Third, extensive experiments are conducted on five datasets for two tasks (open-domain question answering, knowledge-grounded dialogs) and show that JSA-RAG outperforms both vanilla RAG and VRAG, e.g., achieving +4.1% Exact Match on TQA and +10.3% BLEU-4 on DoQA relative over VRAG. Improved retriever performance and low-variance gradients of the posterior retriever are also validated, e.g.,+8.5% R@1 on NQ and +3.7% R@1 on OR-QuAC relative over VRAG.

2. Related Work

Retrieval-Augmented Generation (RAG). RAG [10] has become a widely recognized paradigm for combining parametric memory with nonparametric memory. A major challenge in end-to-end optimization of RAG models is that the optimization needs to marginalize over relevant passages, which are modeled as discrete latent variables with no annotations. Atlas [23] studies some ad-hoc loss functions (including the vanilla RAG loss via TKM) to train the retriever jointly with generator, and does not observe significant systematic differences between the different training objectives. This highlights the need for more principled end-to-end training method, which our JSA-RAG addresses. In addition to investigating new training methods for RAG, there are other research activities around RAG. FiD [24] presents a new strategy to aggregate and combine multiple passages in decoding. In [25], end-to-end training of RAG is applied to specialized domains such as healthcare and news. CoV-RAG [26] integrates a verification module into RAG and uses verification data to finetune RAG generators, while keeping retrievers frozen. RAT [27] combines RAG with chain of thought (CoT) prompting but does not involve any model training. RA-DIT [28] finetunes retrievers and generators separately on a set of multi-task instruction-tuning datasets, which are not end-to-end optimized. *These recent works are orthogonal to and can benefit from JSA-RAG, which focuses on improving end-to-end training of RAG models.*

Learning with discrete latent-variable models. End-to-end training of RAG in essence amounts to learning a discrete latent-variable model. A class of maximum likelihood methods consists of the expectation-maximization (EM) algorithm [29] and its extensions. Variational learning optimizes the Evidence Lower Bound (ELBO) instead of directly maximizing the marginal log-likelihood. VRAG and Hindsight, both based on variational learning, use the TKM approximation to optimize ELBO. RetGen [16] uses the REINFORCE trick [30]. Stochastic RAG [18] uses the Straight-Through trick [31]. These parameter estimators are known to be biased or have high-variance [21]. The JSA algorithm [21,22] is a stochastic extension of the EM algorithm with impressive performance, where both the E-step and the M-step (as they cannot be performed exactly) are extended by the stochastic approximation methodology, hence called joint SA. *To the best of our knowledge, our work is the first to apply and implement the general JSA to successfully realize more powerful and principled end-to-end training method for RAG.*

3. Method: JSA-RAG

3.1. Model

Let (x, y) denote the pair of context and response, both represented by token sequences. Let $\mathcal{K}$ denote the KB, which is

a discrete set of text passages (e.g. Wikipedia chunks). Each passage is also a token sequence. Let h denote the relevant passage in $\mathcal{K}$ needed to generate the response y given the context x, which is treated as a latent variable since there are no annotations. Therefore, we obtain a latent variable model for RAG, with parameters $\theta = (\theta_r, \theta_g)$, which can be decomposed as:

$$p_\theta(y, h|x) = p_{\theta_r}(h|x)p_{\theta_g}(y|x, h) \tag{2}$$

Prior retriever $p_{\theta_r}(h|x)$, is parameterized by θ_r and models the prior relevancy of the passages in $\mathcal{K}$ with respect to the context x. Similar to the original RAG, a bi-encoder architecture for the prior retriever is defined as follows:

$$p_{\theta_r}(h|x) = \frac{\exp\left(\mathbf{e}_\lambda(h)^\top \mathbf{e}_\eta(x)\right)}{\sum_{h' \in \mathcal{K}} \exp\left(\mathbf{e}_\lambda(h')^\top \mathbf{e}_\eta(x)\right)} \tag{3}$$

where $\mathbf{e}_\lambda(x)$ denote the context encoder, parameterized by λ, outputting the dense vector representation (or say, the embedding vector) of the context; $\mathbf{e}_\eta(h)$, the passage encoder, parameterized by η, returns the embedding vector of the passage. We calculate the two embeddings with two separate neural networks, both initialized from BERT [32]. Hence, $\theta_r = (\lambda, \eta)$.

Generator $p_{\theta_g}(y|x, h)$ is parameterized by θ_g and models the sequential generation of the response y given the context x and the passage h. The neural network architecture can be encoder-decoder or decodely-only. In this work, we employ decoder-only LLMs, which calculates the likelihood of the response y as follows:

$$\log p(y|h, x) = \sum_j \log p(y_j|y_{<j}, x, h). \tag{4}$$

where the context x and the retrieved passage h are concatenated to fed into the LLM to generate y.

Posterior retriever is introduced for applying the JSA algorithm to learn the latent variable model Eq. (2). It represents an auxiliary inference model to approximate the posterior probability of selecting passage h when given both context x and reponse y. Similar to the prior retriever, a bi-encoder architecture for the posterior retriever, with parameters $\phi = (\lambda, \xi)$, is defined as follows:

$$q_\phi(h|x, y) = \frac{\exp\left(\mathbf{e}_\lambda(h)^\top \mathbf{e}_\xi(x + y)\right)}{\sum_{h' \in \mathcal{K}} \exp\left(\mathbf{e}_\lambda(h')^\top \mathbf{e}_\xi(x + y)\right)} \tag{5}$$

where the passage encoder $\mathbf{e}_\lambda(h)$ is shared between the prior and the posterior retrievers, but a new BERT based neural network is introduced to calculate the embedding for the combination of context x and response y. In particular, x and y are concatenated, denoted by $x + y$, and are fed to the context-response encoder $\mathbf{e}_\xi(\cdot)$. Note that except for the index rebuilding experiment, all passage encoders are fixed.

Computation consideration. The softmax calculation over the entire KB in Eq. (3) and Eq. (5) for the prior and posterior retrievers are computationally prohibitive even for moderate-sized KBs (e.g., thousands of passages). In practice, we maintain an index on passage embeddings for the KB using FAISS [33]. Given a pair of context and reponse (x, y), we can efficiently retrieve the set of top-k passages under prior and posterior distributions using Maximum Inner Product Search (MIPS) [33], denoted by $\mathcal{S}^{\text{prior}}$ and $\mathcal{S}^{\text{post}}$ respectively (k = 10 in our experiments). The two sets occupy the majority of probabilities for the prior and posterior distributions, and at first thought, can be used to approximate the calculations of Eq. (3) and Eq. (5), respectively. Note that in order to calculate the importance weights for sampled passages used in JSA training (to be clear in Section 3.2 below), we need the prior and posterior probabilities to be calculated over a common set. Therefore, we form a union set by merging $\mathcal{S}^{\text{prior}}$ and $\mathcal{S}^{\text{post}}$, and the softmax calculations in Eq. (3) and Eq. (5) are only taken over this union set.

3.2. Training

Training the RAG model from complete data, i.e., knowing h, can be easily realized by supervised training. For end-to-end training of the RAG model (i.e., conducting unsupervised training without knowing h), we resort to maximizing the marginal likelihood $p_\theta(y|x)$ and applying the JSA algorithm [21, 22].

JSA involves introducing an auxiliary inference model to approximate the intractable posterior $p_\theta(h|x, y)$, which, turns out to take the form of $q_\phi(h|x, y)$, i.e., the posterior retriever. We can jointly train the three components (prior retriever, posterior retriever and generator), which is summarized in Algorithm 1. The JSA algorithm can be viewed as a stochastic extension of the well-known EM algorithm [29] , which iterates Markov Chain Monte Carlo (MCMC) sampling and parameter updating, being analogous to the E-step and the M-step in EM respectively.

E-Step. The sampling step stochastically retrieves passages through sampling from the posterior $p_\theta(h|x, y)$. However, direct sampling from the posterior $p_\theta(h|x, y)$ is intractable, so MCMC sampling is employed. Particularly, using $p_\theta(h|x, y)$ as the target distribution and $q_\phi(h|x, y)$ as the proposal, we sample h through Metropolis independence sampler (MIS) [34] as follows:

1) Propose $h \sim q_\phi(h|x, y)$;

2) Accept h with probability $\min\left\{1, \frac{w(h)}{w(\tilde{h})}\right\}$, where

$$w(h) = \frac{p_\theta(h|x, y)}{q_\phi(h|x, y)} \propto \frac{p_{\theta_r}(h|x)p_{\theta_g}(y|x, h)}{q_\phi(h|x, y)} \tag{6}$$

is the usual importance ratio between the target and the proposal distribution and $\tilde{h}$ denotes the previous value for h along the Markov chain. In practice, we run MIS for several (m) steps, with the chain is initialized from $p_\theta(h|x, y)$.

M-Step. Once we obtain the accepted pseudo labels $\{h^{(1)}, h^{(2)}, ..., h^{(m)}\}$ from MIS, we can treat them as being given. We calculate the gradients for the prior retriever, posterior retriever, and generator models, respectively, and proceed with parameter updating, very similar to the process in supervised training. This is analogous to the M-step in EM, but the proposal q_ϕ is also adapted. In summary, the loss function can be written as:

$$\begin{aligned}\mathcal{L}_{\text{JSA}} = -\frac{1}{m}\sum_{i=1}^{m}\Big(&\log p_{\theta_r}(h^{(i)}|x) + \log p_{\theta_g}(y|x, h^{(i)}) \\ &+ \log q_\phi(h^{(i)}|x, y)\Big)\end{aligned} \tag{7}$$

3.3. Index rebuilding and passage concatenation

Index Rebuilding. In previous work, during training, the index of passage embeddings for the KB is often fixed; therefore, the parameters of the passage encoder (λ) are frozen [7, 10, 35]. In this work, to study whether JSA-RAG can perform end-to-end optimization of all modules - including the passage encoder, we explore an index rebuilding scheme. During training, we no

Algorithm 1 The JSA-RAG algorithm

Require: Training dataset $\mathcal{D} = \{(x, y)\}$, prior retriever $p_{\theta_r}(h|x)$, posterior retriever $q_\phi(h|x, y)$, generator $p_{\theta_g}(y|x, h)$, MIS step number m.
repeat
  Draw a pair of context and response (x, y);
  Monte Carlo sampling:
  Use MIS to draw $\{h^{(1)}, h^{(2)}, ..., h^{(m)}\}$;
  Parameter updating:
  Update θ by ascending:
  $\frac{1}{m}\sum_{i=1}^{m} \nabla_\theta \log \left[p_{\theta_r}(h^{(i)}|x)p_{\theta_g}(y|x, h^{(i)})\right]$;
  Update ϕ by ascending:
  $\frac{1}{m}\sum_{i=1}^{m} \nabla_\phi \log q_\phi(h^{(i)}|x, y)$;
until convergence
return θ and ϕ

longer freeze the parameters of the passage encoder and recalculate the passage embeddings in the index using the updated passage encoder at regular intervals. During passage embedding recalculation, the training process waits; the training is resumed after the index update is completed.

Passage Concatenation. Note that the prior retriever is improved after end-to-end learning. Inspired by FiD [24], we consider a passage concatenation strategy for post-training of the generator while fixing the retriever. The top-k prior retrieved passages are concatenated and append to the context, forming a combined sequence that is fed into the generator for response generation. In this way, the generator is post-trained and in the same way, the generator is used in decoding. This shows the bonus offered by JSA-RAG.

Note that the above two methods are only used in the experiments described in Section 4.5.

3.4. Decoding

During testing, we use "Top-k Documents Decoding", following VRAG [7], with k = 10. Specifically, given a context x, we employ the trained prior retriever to fetch the top-k passages $\{h^{(1)}, \cdots, h^{(k)}\}$. The context x and the retrieved passage $h^{(i)}$ are concatenated and fed into the generator, a beam search is performed to generate the top response $y^{(i)}$, $i = 1, \cdots, k$. We estimate $p(y^{(i)}|x)$ using the product of two terms: $p(y^{(i)}|x) \approx p(h^{(i)}|x)p(y^{(i)}|h^{(i)}, x)$. Finally, we select the response $y^{(i)}$ with the highest estimated probability as the final output for the given context x. This is a simplified "Fast Decoding" in [10] and performs well in our experiments.

4. Experiment

To evaluate the effectiveness of the JSA-RAG method, we use VRAG and RAG as baseline end-to-end methods. The evaluation is taken on two tasks - open-domain question answering and knowledge-grounded dialogs. Comprehensive experiments are conducted to evaluate the performance of JSA-RAG, focusing on aspects such as generation quality and retrieval recall. Ablation studies are also conducted to analyze the effects of JSA-RAG in controlling gradient variance and optimizing retrieval efficiency. Specifically, we compare the performance of the posterior retrievers in JSA-RAG and VRAG, as well as fluncations in gradient norms. Furthermore, we investigate the effect of index rebuilding in training. We study the passage concatenation strategy for post-training of the generator while fixing the retriever. These further demonstrate the capability and bonus offered by JSA-RAG.

4.1. Datasets

Open-domain question answering (ODQA) requires extensive external knowledge to answer questions, which is the primary task explored with RAG systems. We mainly consider three ODQA datasets: NaturalQuestions (NQ) [36], TriviaQA (TQA) [37], and MS-MARCO [38][1]. For NQ and TQA, we use the Wikipedia December 2018 dump, which contains a total of 24M chunks (passages); for MS-MARCO, instead of using the 10 provided reference passages, we extract its QA pairs and use the MS-MARCO passages from TREC2019 as the KB [38].

Knowledge grounded dialogs. In dialog datasets, we use conversation history turns as x to retrieve relevant passages and take the response of the current turn as y, thus constructing (x, y) pairs. We use the OR-QuAC [39] and the DoQA [40] datasets. OR-QuAC is an open-domain dialog question answering dataset derived from the QUAC (Question Answering in Context) corpus, requiring models to retrieve and reason over external knowledge to answer multi-turn conversational questions. DoQA comprises of open-ended dialog conversations on different domains like cooking, travel and movies. Both datasets follow the knowledge base settings used in VRAG [7]. For OR-QuAC, its KB contains 68k passages, while for DoQA, its KB contains 1.2k passages.

4.2. Experiment settings

4.2.1. Model and evaluation settings

In open-domain QA tasks, we employ BGE-large (326M parameters) [12] to initialize both the prior and posterior retrievers and use Mistral-7B [41] as the generator, which is fine-tuned with LoRA [42,43] during training. To evaluate end-to-end generation, we use Exact Match for NQ and TQA, and BLEU-1 and Rouge-L for MS-MARCO. For retriever performance, we use Recall@1 and Recall@10 to measure the accuracy of retrieved results for all three datasets.

For knowledge-grounded dialog tasks, we initialize the prior and posterior retrievers with DPR (124M parameters) [5] and use GPT-2 (124M parameters) [44] as the generator. For end-to-end generation evaluation, we use BLEU-1, BLEU-4 [45], and F1; for retriever performance, we employ metrics the same as in open-domain QA.

For all experiments, top-10 retrieval is used for both the prior and posterior retrievers during training and testing.

4.2.2. Gold passage annotation for retriever evaluation

In testing retrieval performance, some datasets such as NQ, TQA, and MS-MARCO do not have annotations for gold passages in their corresponding KBs. A gold passage refers to a passage containing information capable of answering a question. To obtain such passages or to find them as closely as possible, we first perform a posterior retrieval based on BGE-large to retrieve 100 relevant passages from the knowledge base for each question (the posterior retriever performs retrieval using question-answer pairs to incorporate more information). We then prompt GPT-4o to select the one passage that it deems the most capable of answering the question from the 100 candidates. The specific prompt is shown in Table 1. It should be

[1]The MS-MARCO dataset has two versions, and the version we use in this work is MS-MARCO v1.

Table 1: *Prompt for gold passage selection via GPT-4o for retriever evaluation*

Prompt Text for Gold Passage Selection
`Question: {question}, Provided Answers: {answers}. Please select the ID of the passage that best answers the question from the following paragraphs. If there is no passage you think can generate the correct answer, select the ID of the passage that comes closest to answering the question.` **`Note!!! Only return the value of the passage's id key.`**

Table 2: *LLM prompt templates used in evaluation for different tasks*

Tasks	**LLM Prompt Template**
ODQA	`[INST] Give a short answer to the Question based on relevant information given in Input. \nInput:{retrieved passage} \nQuestion: {q} \n[/INST]{answer}`
Dialogs	`Input:{retrieved passage}\n <speaker1>{turn1} <speaker2>{turn2} <speaker1>{turn3} <speaker2>{answer}`

noted that these GPT-4o-generated annotations are never used in model training, but only used in testing to evaluate the prior retrievers (Table 5) and the posterior retrievers (Table 6), for those datasets without gold passage annotations (i.e., NQ, TQA and MS-MARCO in our experiments).

4.2.3. Index server

Technically, we construct an FAISS index for fast retrieval and deploy it as a standalone server. This allows the main program to perform retrieval via API calls. This setup optimizes GPU memory utilization: by eliminating the need to pre-reserve GPU memory for index loading, the main program can allocate more dedicated VRAM to model computations. Notably, the index persists across experiments (when different experiments use the same Wikepedia KB), eliminating redundant embedding recomputation and significantly reducing training time.

4.2.4. Training details

In our experiments, all methods were run for 20,000 steps with batch size being 1 and the best results were recorded. During training, we set different learning rates for the retriever and generator. The loss was optimized using the AdamW optimizer. For the dialog task, using GPT-2 and DPR models, the learning rates for the generator and retriever were set to 1×10^{-5} and 1×10^{-5}, respectively. For the ODQA task with Mistral-7B and BGE models, the learning rates for the generator and retriever were 2×10^{-5} and 1×10^{-5}, respectively. In the passage concatenation experiment, only the generator was post-trained at a learning rate of 1×10^{-5} for 10,000 steps. Additionally, the hyperparameter involved in JSA include: MIS sampling steps $m = 50$.

Table 3: *LoRA Hyperparameter Settings*

Parameter	**Value**
Task Type	Causal Language Modeling (CAUSAL_LM)
Rank Reduction Factor (r)	8
LoRA Scaling Factor (α)	16.0
Dropout Probability	0.0
Bias Training Strategy	None
Target Modules	`k_proj`, `q_proj`, `v_proj`, `o_proj`, `gate_proj`, `down_proj`, `up_proj`

For training with the dialog datasets, we used 8 NVIDIA 3090 GPUs with 24GB VRAM for both training and index storing. For ODQA experiments, training was conducted on 8 A100 GPUs with 40GB VRAM, where 4 GPUs were dedicated to main training and the other 4 to building the index server.

GPT-2 was fine-tuned with full parameters, while the Mistral-7B model was wrapped with a PEFT model for LoRA fine-tuning. The configuration of Low-Rank Adaptation (LoRA) parameters used in this study is presented in Table 3. These settings were applied uniformly across all experiments unless otherwise specified.

4.2.5. LLM prompt template

During training, to enable the generator to more clearly combine the context and the retrieved passages, we employ different LLM prompt templates for different tasks, as shown in Table 2.

4.3. Main result

Based on the results in Table 4, JSA-RAG demonstrates significant superiority on dialog datasets (DoQA and OR-QUAC) and open-domain QA tasks, with all evaluated metrics outperforming baselines (vanilla RAG and VRAG).

End-to-end generation performance. On DoQA, JSA-RAG achieves a BLEU-4 score of 17.11 (+10.3% relative over VRAG) and an F1 score of 27.84 (+6.9% relative over VRAG), and on OR-QUAC, its F1 score of 18.41 represents a relative 4.4% gain over VRAG, highlighting superior contextual knowledge integration for complex dialog reasoning. In open-domain QA, JSA-RAG excels in TQA with an Exact Match score of 75.23 (+4.1% relative over VRAG) and NQ with an Exact Match score of 51.05 (+4.1% relative over VRAG), indicating robust handling of multi-step questions, and in MS-MARCO with a Rouge-L score of 37.96 (+3.4% relative over VRAG), reflecting fluent and contextually faithful generation. JSA-RAG shows significant improvements on the NQ dataset as well. From these results, we can observe that JSA-RAG completely outperforms the other two methods in end-to-end generation.

Retrieval performance. Going beyond end-to-end generation performance, we aim to explore whether the JSA-RAG promotes joint improvement of retriever and generator in end-to-end training. Thus, we conduct experiments to evaluate the retrieval performance. On Table 5, JSA-RAG demonstrates consistent improvements in prior retriever performance across dialogs (OR-QUAC, DoQA) and open-domain QA (NQ, TQA, MS-MARCO) datasets. This reveals that the retrievers can get enhanced in JSA-RAG end-to-end training. On OR-QUAC,

Table 4: *Performance comparison of different methods on knowledge-grounded dialog and open-domain question answering datasets.*

Method	Knowledge-Grounded Dialog						Open-Domain Question Answering			
	DoQA			OR-QUAC			NQ	TQA	MS-MARCO	
	BLEU-4	BLEU-1	F1	BLEU-4	BLEU-1	F1	EM	EM	BLEU-1	Rouge-L
RAG	15.39	21.69	25.91	6.57	13.51	17.28	50.52	72.82	34.23	36.54
VRAG	15.51	21.55	26.02	6.71	13.87	17.63	49.03	72.26	34.14	36.70
JSA-RAG	**17.11**	**23.36**	**27.84**	**7.76**	**14.59**	**18.41**	**51.05**	**75.23**	**35.28**	**37.96**

Table 5: *Performance evaluation of the prior retrievers for different methods. The base retrievers for knowledge-grounded dialog datasets and open-domain question answering datasets are BGE-large and DPR, respectively.*

Method	Knowledge-Grounded Dialog						Open-Domain Question Answering								
	OR-QUAC			DoQA			NQ			TQA			MS-MARCO		
	R@1	R@10	MRR@10	R@1	R@10	MRR@10	R@1	R@10	MRR@10	R@1	R@10	MRR@10	R@1	R@10	MRR@10
Base	31.16	77.74	48.28	58.59	83.13	66.94	26.25	63.73	37.64	33.33	66.73	44.21	10.51	36.81	17.91
RAG	38.97	83.35	55.93	67.61	87.33	74.74	27.58	66.97	39.96	36.01	69.48	46.19	23.17	68.66	36.48
VRAG	38.91	83.48	55.98	68.01	87.49	74.78	26.95	64.04	38.21	36.81	70.07	46.73	23.68	68.81	37.53
JSA-RAG	**40.35**	**84.66**	**57.97**	**68.09**	**87.57**	**75.06**	**29.23**	**67.27**	**41.04**	**37.39**	**70.67**	**46.90**	**24.75**	**71.32**	**38.65**

JSA-RAG achieves the highest R@1 (40.35, +3.7% relative over VRAG), R@10 (84.66, +1.4% relative over VRAG) and MRR@10 (57.97, +3.6% relative over VRAG), reflecting successful optimizations for dialog retrieval. In open-domain QA, JSA-RAG outperforms baselines on NQ (R@1: 29.23 +8.5%, R@10: 67.27 +5.0%, MRR@10:41.04 +7.4% relative over VRAG), TQA (R@1: 37.39, +1.5% relative over VRAG) and MS-MARCO (R@1: 24.75 +4.5%, R@10:71.32 +3.6% relative over VRAG), indicating stronger retrieval of relevant passages. The superiority of JSA-RAG in R@1 and R@10 across multiple datasets indicates that passages filtered by MIS are more helpful in guiding the training of retrievers, compared to simply using a posterior retriever to fetch top-10 passages.

Comparative analysis. By combining the analysis of end-to-end generation performance and retriever performance, we find that JSA-RAG comprehensively outperforms both RAG and VRAG. Notably, on TQA and MS-MARCO, although VRAG introduces the posterior retriever and improves retriever performance, its generation performance declines compared to RAG. This exhibits asynchrony in end-to-end optimization, where retriever performance improves while generator performance decreases instead. In contrast, JSA-RAG enables more effective joint optimization between retrievers and generators, achieving simultaneous improvements in both retriever accuracy (e.g., higher R@1, R@10, MRR@10) and generative quality (e.g., superior BLEU-4, F1 scores). This demonstrates that the knowledge pieces selected by JSA-RAG's MIS step can not only enhance retriever performance but also align well with the preferences for generating response, rather than merely maximizing the relevance scores of retrieved knowledge pieces.

4.4. Ablations

We conduct ablation experiments on the posterior retriever from two aspects to help understand intuitively why JSA-RAG outperforms VRAG. First, we analyze the performance of the posterior retriever. We test the trained posterior retriever on recall@1, recall@10, and MRR@10 metrics using the OR-QUAC dataset, a moderately scaled dataset with gold passage annotations. Second, we monitor the gradient variation of the posterior retriever during training. This allows us to observe how the gradients fluctuate along the training steps to intuitively compare the gradient variance between JSA and VRAG.

Table 6: *Performance evaluation of the posterior retrievers on the OR-QuAC dataset.*

Dataset	Method	R@1	R@10	MRR@10
OR-QuAC	DPR	44.32	90.72	63.88
	VRAG	45.66	91.26	64.52
	JSA-RAG	**46.91**	**91.43**	**65.42**

Performance of posterior retriever. As shown in Table 6, on the QuAC dataset, JSA-RAG's posterior retriever outperforms VRAG across all evaluated metrics: R@1 (46.91 vs. 45.66), R@10 (91.43 vs. 91.26), and MRR@10 (65.42 vs. 64.52), achieving improvements of 2.7%, 0.2%, and 1.4% respectively. These results indicate that JSA-RAG's posterior retriever captures relevant knowledge pieces more accurately. Presumably, this is because the low-variance gradients allow the posterior retriever to converge more efficiently toward the true posterior distribution during training. Similarly, better quality of passages retrieved by MIS also benefit posterior retriever.

Variance in gradient norm. As shown in Figure 2, we record the gradient norms of the posterior retrievers for JSA-RAG and VRAG every 50 steps during the first 4,000 training steps. The gradient norms of JSA are of low variance, while the gradient norms of VRAG frequently exhibit "sharp" spikes. This observation confirms that during training, JSA-RAG provides gradients with lower variance, enabling more stable training dynamics and thus yielding higher performance.

4.5. Experiments on index rebuilding and passage concatenation

To further examine the practical behavior of JSA-RAG beyond the main end-to-end training results, we conduct two additional studies on index rebuilding and passage concatenation. The first study evaluates whether periodically refreshing the re-

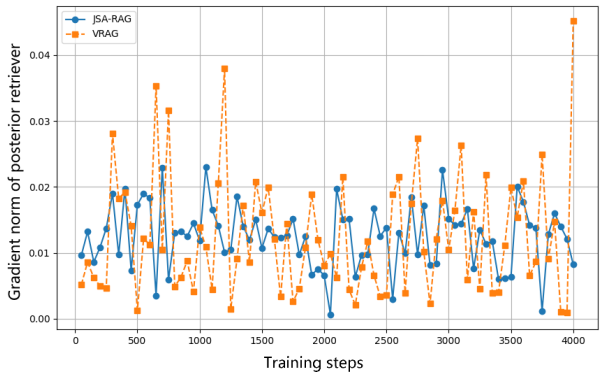

Figure 2: *Comparison of the gradient norms from the posterior retriever.*

trieval index can better align the evolving passage encoder with the stored passage representations during training. The second study investigates whether a post-training stage with concatenated passages can further improve generation when the retriever is fixed. Together, these experiments provide a more comprehensive view of the robustness, scalability, and extensibility of the proposed framework.

Index rebuilding. In the index rebuilding experiment, unlike the main experiment, we do not freeze the parameters of the passage encoder. Every 100 steps, we recalculate the passage embeddings using the updated passage encoder and rebuild the index. As Table 7 shows, updating the index on OR-QuAC improves all metrics for all methods, with substantial gains in retriever performance, and JSA-RAG remains the top performance. These findings show that JSA-RAG is able to enhance the performance of all system components in an end-to-end training framework. Meanwhile, it should be noted that in our experiments, model training and index rebuilding are serially conducted. In the OR-QuAC (68k passages) experiment, it takes about 3 minutes for 100 training steps, and then index rebuilding takes less than 2 minutes, which hardly affects the training time. However, for larger scale experiments with Wikipedia (24M packages), index rebuilding takes 6 hours with the hardware in our experiments (4*A100 (40G)). When we used a long interval for index rebuiding (we tried 2000-5000 steps in experiments), marginal performance improvements were obtained compared to the results of not using index rebuilding, since the model training behavior does not change much. However, if we reduce the interval for index rebuilding (e.g., every 100 steps), then it will add 200×6 hours with huge time cost (for a total of 20,000 steps for one experiment). Asynchronous algorithm research (parameter training and index rebuilding are performed in parallel) is interesting future work. *It should be noted that such asynchronous algorithm is orthogonal to the core contribution of this work – the developement of JSA-RAG beyond the prior arts which are vanilla RAG (using top-K marginalization) and variational approaches like VRAG.*

Passage concatenation. In the passage concatenation strategy, we freeze all parameters except the generator. As shown in Table 8. On the dataset NQ,TQA and MS-MARCO, post-training with the passage concatenation strategy is found to improve performance across all methods. After all methods are subjected to post-training, JSA-RAG still significantly outperforms RAG and VRAG. This robustness shows the practical applicability of the JSA-RAG approach.

Table 7: *Performance of index rebuilding on OR-QuAC (index rebuilt every 100 steps during training).*

Method	Rebuild	BLEU-4	R@1	R@10	MRR@10
RAG	✗	6.57	38.97	83.35	55.93
	✓	8.50	47.77	89.75	65.39
VRAG	✗	6.71	38.91	83.48	55.98
	✓	8.58	48.54	90.09	65.90
JSA-RAG	✗	7.76	40.35	84.66	57.97
	✓	**10.26**	**49.44**	**90.11**	**66.36**

Table 8: *Performance of passage concatenation on ODQA (top 10 passages concatenated with context).*

Method	Concat	NQ	TQA	MS-MARCO	
		EM	EM	BLEU-1	Rouge-L
RAG	✗	50.52	72.82	34.23	36.54
	✓	51.10	74.84	34.00	37.31
VRAG	✗	49.03	72.26	34.14	36.70
	✓	51.99	75.54	34.81	37.86
JSA-RAG	✗	51.05	75.23	35.28	37.96
	✓	**52.35**	**76.11**	**35.61**	**38.81**

4.6. Computation cost in training

To evaluate the computational efficiency of different methods, we measure the training time for 100 iteration steps on the OR-QuAC dataset (batch size = 1). On OR-QuAC, 100 iterations take 148s for RAG, 153s for VRAG, and 198s for JSA-RAG. Thus JSA-RAG is somewhat slower, mainly due to MIS sampling, but still comparable to the baselines and gives consistent gains.

5. Conclusion and Future Work

A major challenge in end-to-end optimization of the RAG model is that the optimization needs to marginalize over relevant passages from a knowledge base, which are modeled as discrete latent variables with no annotations. Traditional top-K marginalization and variational RAG (VRAG) suffer from biased or high-variance gradient estimates. In this paper, we propose and develop joint stochastic approximation (JSA) based end-to-end training of RAG, which is referred to as JSA-RAG. The JSA algorithm is a stochastic extension of the EM algorithm and is particularly powerful in estimating discrete latent variable models. JSA-RAG achieves substantial improvements across multiple tasks and datasets, compared to vanilla RAG and VRAG. Notably, it can be seen from Section 4.6 that the training time cost of JSA-RAG is comparable to RAG and VRAG. Beyond performance evaluation, we demonstrate that JSA-RAG exhibits lower gradient variance than VRAG. We also conduct extensive investigations to further strengthen the JSA-RAG framework, including the index rebuilding and the passage concatenation strategies.

Remarkably, the potential advantage of the JSA-RAG approach in learning discrete latent variable models suggests promising directions for future research, particularly in learning multi-step agents. Basically, RAG can be viewed as a two-step agent (retrieve-then-generate). Currently, the multi-step trajec-

tory of thinking, reasoning, tool use, and planning in building agents needs to be synthesized or annotated. The JSA-RAG methodology investigated in this paper can be extended to learning such multi-step agents. This avenue of exploration is highly promising and warrants further investigation.

References

[1] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, and A. Miller, "Language models as knowledge bases?" in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 2463–2473.

[2] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[3] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading wikipedia to answer open-domain questions," pp. 1870–1879, 2017.

[4] K. Lee, M.-W. Chang, and K. Toutanova, "Latent retrieval for weakly supervised open domain question answering," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 6086–6096.

[5] 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)*. Association for Computational Linguistics, 2020.

[6] B. Kim, J. Ahn, and G. Kim, "Sequential latent knowledge selection for knowledge-grounded dialogue."

[7] M. Mishra, D. Madan, G. Pandey, and D. Contractor, "Variational learning for unsupervised knowledge grounded dialogs," in *International Joint Conference on Artificial Intelligence*, 2022.

[8] Y. Cai, H. Liu, Z. Ou, Y. Huang, and J. Feng, "Knowledge-retrieval task-oriented dialog systems with semi-supervision," in *Proc. Interspeech 2023*, 2023, pp. 4673–4677.

[9] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, "Retrieval augmented language model pre-training," in *International conference on machine learning*. PMLR, 2020, pp. 3929–3938.

[10] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.

[11] G. Izacard, M. Caron, L. Hosseini, S. Riedel, P. Bojanowski, A. Joulin, and E. Grave, "Unsupervised dense information retrieval with contrastive learning," *Transactions on Machine Learning Research*.

[12] P. Zhang, S. Xiao, Z. Liu, Z. Dou, and J.-Y. Nie, "Retrieve anything to augment large language models," *arXiv preprint arXiv:2310.07554*, 2023.

[13] O. Khattab, K. Santhanam, X. L. Li, D. Hall, P. Liang, C. Potts, and M. Zaharia, "Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp," *arXiv preprint arXiv:2212.14024*, 2022.

[14] T. Zhang, S. G. Patil, N. Jain, S. Shen, M. Zaharia, I. Stoica, and J. E. Gonzalez, "Raft: Adapting language model to domain specific rag," in *First Conference on Language Modeling*, 2024.

[15] Y. Zhao, H. Cao, X. Zhao, and Z. Ou, "An empirical study of retrieval augmented generation with chain-of-thought," in *2024 IEEE 14th International Symposium on Chinese Spoken Language Processing (ISCSLP)*. IEEE, 2024, pp. 436–440.

[16] Y. Zhang, S. Sun, X. Gao, Y. Fang, C. Brockett, M. Galley, J. Gao, and B. Dolan, "Retgen: A joint framework for retrieval and grounded text generation modeling," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 10, 2022, pp. 11 739–11 747.

[17] G. Han, D. Jo, D. Nam, E. Yoon, T. Kwon, S. Rho, K.-W. On, C. Yoo, and S. Kim, "Efficient latent variable modeling for knowledge-grounded dialogue generation," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 2683–2702.

[18] H. Zamani and M. Bendersky, "Stochastic rag: End-to-end retrieval-augmented generation through expected utility maximization," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024, pp. 2641–2646.

[19] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations (ICLR)*, Y. Bengio and Y. LeCun, Eds., 2014.

[20] A. Paranjape, O. Khattab, C. Potts, M. Zaharia, and C. D. Manning, "Hindsight: Posterior-guided training of retrievers for improved open-ended generation," in *International Conference on Learning Representations*.

[21] Z. Ou and Y. Song, "Joint stochastic approximation and its application to learning discrete latent variable models," in *Conference on Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 929–938.

[22] H. Xu and Z. Ou, "Joint stochastic approximation learning of Helmholtz machines," in *ICLR Workshop Track*, 2016.

[23] G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, and E. Grave, "Atlas: Few-shot learning with retrieval augmented language models," *Journal of Machine Learning Research*, vol. 24, no. 251, pp. 1–43, 2023.

[24] G. Izacard and É. Grave, "Leveraging passage retrieval with generative models for open domain question answering," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2021, pp. 874–880.

[25] S. Siriwardhana, R. Weerasekera, E. Wen, T. Kaluarachchi, R. Rana, and S. Nanayakkara, "Improving the domain adaptation of retrieval augmented generation (rag) models for open domain question answering," *Transactions of the Association for Computational Linguistics*, vol. 11, pp. 1–17, 2023.

[26] B. He, N. Chen, X. He, L. Yan, Z. Wei, J. Luo, and Z.-H. Ling, "Retrieving, rethinking and revising: The chain-of-verification can improve retrieval augmented generation," *arXiv preprint arXiv:2410.05801*, 2024.

[27] Z. Wang, A. Liu, H. Lin, J. Li, X. Ma, and Y. Liang, "Rat: Retrieval augmented thoughts elicit context-aware reasoning and verification in long-horizon generation," in *NeurIPS 2024 Workshop on Open-World Agents*, 2024.

[28] X. V. Lin, X. Chen, M. Chen, W. Shi, M. Lomeli, R. James, P. Rodriguez, J. Kahn, G. Szilvasy, M. Lewis *et al.*, "Ra-dit: Retrieval-augmented dual instruction tuning," in *The Twelfth International Conference on Learning Representations*, 2023.

[29] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, vol. 39, 1977.

[30] J. Paisley, D. M. Blei, and M. I. Jordan, "Variational bayesian inference with stochastic search," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, 2012, pp. 1363–1370.

[31] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[32] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

[33] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[34] J. S. Liu, *Monte Carlo strategies in scientific computing*. Springer, 2001, vol. 10.

[35] X. V. Lin, X. Chen, M. Chen, W. Shi, M. Lomeli, R. James, P. Rodriguez, J. Kahn, G. Szilvasy, M. Lewis *et al.*, "Ra-dit: Retrieval-augmented dual instruction tuning," in *The Twelfth International Conference on Learning Representations*, 2023.

[36] 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*, vol. 7, pp. 453–466, 2019.

[37] 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)*, 2017, pp. 1601–1611.

[38] P. Bajaj, D. Campos, N. Craswell, L. Deng, J. Gao, X. Liu, R. Majumder, A. McNamara, B. Mitra, T. Nguyen *et al.*, "Ms marco: A human generated machine reading comprehension dataset," *arXiv preprint arXiv:1611.09268*, 2016.

[39] C. Qu, L. Yang, C. Chen, M. Qiu, W. B. Croft, and M. Iyyer, "Open-retrieval conversational question answering," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 539–548.

[40] J. A. Campos, A. Otegi, A. Soroa, J. M. Deriu, M. Cieliebak, and E. Agirre, "Doqa-accessing domain-specific faqs via conversational qa," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 7302–7314.

[41] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de Las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7b," *ArXiv*, vol. abs/2310.06825, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:263830494

[42] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models." *ICLR*, vol. 1, no. 2, p. 3, 2022.

[43] Z. Han, C. Gao, J. Liu, J. Zhang, and S. Q. Zhang, "Parameter-efficient fine-tuning for large models: A comprehensive survey," *Transactions on Machine Learning Research*.

[44] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[45] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics (ACL)*. Association for Computational Linguistics, 2002, pp. 311–318.