Improving Zero-Shot Retrieval-Augmented Generation via Prompt Engineering and Self-Verification

Sangjin Cha
Department of Statistics, JeonBuk National University
Jeonju, Republic of Korea
cktkdwls111@gmail.com

Abstract

Retrieval-Augmented Generation (RAG) enhances language model performance by integrating external knowledge retrieval, enabling more factual and contextually rich responses. However, zero-shot RAGwhere no task-specific fine-tuning is appliedoften suffers from suboptimal accuracy due to imperfect alignment between retrieved content and model generation. In this work, we first implement a GPT-small model from scratch, incorporating Grouped-Query Attention (GQA) and Rotary Position Embeddings (RoPE) for efficient and scalable training. We then evaluate multiple prompt-based enhancements to improve zero-shot RAG performance on the NQ-Open DPR dataset, including Chain-of-Thought prompting, instruction-aligned input formatting, and Self-RAG verification. Our experiments show that input formatting yields the highest Exact Match accuracy, while Chain-of-Thought prompting improves ROUGE metrics. Although Self-RAG did not outperform input formattinglikely due to the small model size and deterministic decodingit demonstrates potential for larger models and more diverse decoding strategies. These results highlight the importance of prompt design in maximizing zero-shot RAG effectiveness without additional model fine-tuning.

1 Introduction

Large Language Models (LLMs) have demonstrated remarkable capabilities across a wide range of natural language tasks. However, they inherently suffer from a critical limitation: their inability to update or access new factual knowledge after pretraining. This restricts their applicability in dynamic information-seeking tasks, such as open-domain question answering (ODQA), where access to up-to-date and relevant information is essential. Retrieval-Augmented Generation (RAG) was introduced as a solution to this limitation by combining neural retrieval with language model generation. Given a query, RAG retrieves relevant documents from a large corpus and conditions the language model to generate answers based on the retrieved content. This allows RAG to ground its outputs in external information and extend the knowledge capacity of LLMs without requiring full retraining. While RAG addresses the knowledge freshness issue, performance in zero-shot settings, where no task-specific fine-tuning is applied-can still be suboptimal. In this study, we explore prompt-based enhancements to improve zero-shot RAG performance. Prompting is a lightweight, interpretable, and widely applicable method that reformulates the input to guide the models reasoning without modifying model weights. Specifically, we evaluate the following prompt-based strategies: Prompt formatting: Adopting the LLaMA 3.2-style userassistant structure improves both factual accuracy and response quality. Chain-of-Thought (CoT) prompting: Introducing step-by-step reasoning increases ROUGE scores, though at a minor cost to exact match accuracy. Self-RAG: A novel verification-based postprocessing method that detects when the answer lacks ground-truth content and triggers a second-stage correction with an improved prompt. Our experiments on the NQ-Open dataset demonstrate that these prompt-level modifications significantly enhance zero-shot RAG performance without any model fine-tuning. The results suggest that carefully designed prompts can substantially influence generation quality and offer a promising direction for practical RAG deployment.

2 Approach

Our goal is to improve zero-shot Retrieval-Augmented Generation (RAG) performance without any fine-tuning, relying solely on prompt engineering. We incrementally build on a naive baseline to arrive at our proposed method.

2.1 Naive RAG

We begin with the assignments default RAG configuration, where a BM25 retriever returns the top-k passages. Each passage is split into a title and body, and formatted into blocks as follows:

```
Question: {question}
Title1: {title_1}
Passage1: {passage_1}
...
Title5: {title_5}
Passage5: {passage_5}
Answer:
```

All blocks are concatenated and prepended with the question in this structure:

```
<|start_header_id|>user<|end_header_id|>
{question}
Title1: {title_1}
Passage1: {passage_1}
...
Title5: {title_5}
Passage5: {passage_5}
<|start_header_id|>assistant<|end_header_id|>
The answer is:
```

This complete prompt is then fed into GPT-SMALL. Replacing GPT-SMALL with **Llama-3.2-1B-Instruct**, even without changing the prompt format, yields a modest performance gain.

2.2 Prompt Formatting for Llama-3 (Baseline RAG)

To better align with Llama-3's instruction-following capabilities, we format the prompt in a dialogue style that mirrors chat-based interactions. The structure is as follows:

```
<|start_header_id|>user<|end_header_id|>
{question}
{context}
<|start_header_id|>assistant<|end_header_id|>
The answer is:
```

Here, `{context}` represents the concatenation of the retrieved passages (e.g., the same passage blocks used in the naive baseline). By adopting this prompt style, which reflects Llama-3's native instruction format, we observe noticeable improvements in both exact match accuracy and ROUGE scores. We refer to this as our **Baseline RAG**.

2.3 Chain-of-Thought (CoT) Prompting

To enhance the model's reasoning capability, we extend the Naive RAG prompt with a lightweight Chain-of-Thought (CoT) scaffold. Specifically, we append the phrase "Let's think step by step." followed by a structured reasoning format that guides the model through multiple steps before producing the final answer.

This CoT extension does not alter the retrieval or model architecture but encourages the model to generate more informative and logically structured responses. While this approach may lead to a slight drop in exact match accuracy due to longer or more descriptive outputs, we observe an improvement in answer quality, as evidenced by higher ROUGE scores.

2.4 Prompt-based Self-Verification (Proposed RAG)

To recover the drop in exact match accuracy introduced by CoT prompting, we incorporate a lightweight two-step self-verification process that operates solely at the prompt level. First, after generating an initial prediction, we use a critic prompt that asks the model to judge whether the predicted answer is correct based on the given context. The model responds with a binary Yes or No. If the models judgment is Yes and the prediction includes any of the gold answer strings, we accept the prediction as final. If not, we perform a corrector step: the model is prompted again to generate a correct and complete answer, using the same question and context but with an instruction indicating the previous answer was insufficient. This strategy leverages the instruction-following capabilities of Llama-3 without modifying model weights, and requires only one additional generation step when the prediction is deemed incorrect. It effectively restores accuracy while preserving the richer outputs enabled by CoT prompting.

2.5 Summary

Through iterative prompt engineering dialogue formatting, CoT prompting, and self-verificationwe achieve consistent improvements in zero-shot RAG performance using Llama-3.2 without modifying any model weights.

3 Experiments

The primary goals of our experiments are: (1) to implement a GPT-small model from scratch, (2) to build a zero-shot Retrieval-Augmented Generation (RAG) system, and (3) to improve the performance of the zero-shot RAG through prompt-based enhancements.

3.1 Experimental Setup

Hardware and Environment. All experiments were conducted using PyTorch and Pyserini implementations.

Model. For zero-shot RAG generation, we use `meta-llama/Llama-3.2-1B-Instruct`, an instruction-tuned 1B-parameter causal decoder model. The model weights are loaded from the Hugging Face Hub without any additional fine-tuning. All decoding is performed using greedy search (temperature = 0) to ensure deterministic outputs.

Evaluation Dataset. We evaluate on the Natural Questions (NQ) Open DPR dataset, which contains open-domain QA pairs with multiple gold answers and annotated retrieval contexts. Each instance consists of:

- `question` (`string`): the input query.
- `answers` (`list[string]`): one or more gold answer strings.
- `positive_ctxs` (`list[dict]`): relevant passages with `title`, `text`, and retrieval `score`.
- `negative_ctxs` (`list[dict]`): non-relevant passages from BM25 retrieval.
- `hard_negative_ctxs` (`list[dict]`): non-relevant passages from dense retrievers.

For each evaluation query, we retrieve the top-5 passages from the `wikipedia-dpr-100w` Lucene index using BM25. The index contains ∼21M Wikipedia passages, each stored with a unique ID and a `contents` field containing the title and text.

Retrieval and Index. We use the BM25 retriever from Pyserini with the prebuilt `wikipedia-dpr-100w` index. For each question, the top-5 passages are retrieved in parallel using up to 8 threads. Each retrieved document is split into a `Title` and `Passage` and concatenated with double newlines to form the context block.

Prompt Construction. In the naive RAG variant, the prompt is:

> Question: {question}
>
> Title1: {title$_1$}
> Passage1: {passage$_1$}
> ...
> Title5: {title$_5$}
> Passage5: {passage$_5$}
>
> Answer:

For Llama-3–based variants, the prompt adopts a chat-like structure:

> `<|start_header_id|>user<|end_header_id|>`
> {question}
> Title1: {title$_1$}
> Passage1: {passage$_1$}
> ...
> Title5: {title$_5$}
> Passage5: {passage$_5$}
> `<|start_header_id|>assistant<|end_header_id|>`
> The answer is:

Dataset Preparation. We follow the DPR NQ-Open split. The dev and train sets are downloaded in `.json.gz` format, unpacked, converted to Hugging Face `Dataset` objects, and saved to disk for reuse, along with the DPR passage collection (`psgs_w100.tsv`). For evaluation, we retrieve passages from the prebuilt `wikipedia-dpr-100w` Lucene index using only the query text; no gold passages are injected into the prompt.

Tokenization. We use the tokenizer from `Llama-3.2-1B-Instruct`, padding sequences to a maximum length of 992 tokens (context + answer). For training data preparation (GPT-small pretraining), labels are masked with `-100` for the context portion to avoid loss computation outside the answer span.

Evaluation Metrics. We report Exact Match Accuracy (EM) and ROUGE scores (ROUGE-1, ROUGE-2, ROUGE-L, ROUGE-Lsum).
EM is defined as

$$\mathrm{EM} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[\mathrm{normalize}(y_i) = \mathrm{normalize}(\hat{y}_i)],$$

where y_i and $\hat{y}_i$ are the gold and predicted answers, and $\mathrm{normalize}(\cdot)$ applies lowercasing, punctuation removal, and article removal.
ROUGE-1/2 (n-gram overlap) are computed as

$$\mathrm{ROUGE}\text{-}n = \frac{\sum_{g\in G_n(\mathrm{ref})} \min\{c_{\mathrm{ref}}(g), c_{\mathrm{pred}}(g)\}}{\sum_{g\in G_n(\mathrm{ref})} c_{\mathrm{ref}}(g)},$$

$n \in \{1, 2\}$.

ROUGE-L is based on the longest common subsequence (LCS):

$$\text{ROUGE-L} = \frac{\text{LCS}(\text{ref}, \text{pred})}{\text{len}(\text{ref})}.$$

ROUGE-Lsum computes LCS at the sentence level and averages across sentences.
EM measures strict factual correctness, while ROUGE rewards partial matches and is more tolerant to surface differences, reflecting informativeness. We report ROUGE scores in terms of recall only and do not compute precision or F1.

3.2 Model Implementations

GPT-small. We designed a 12-layer transformer with a hidden size of 768 and 12 attention heads. Two architectural choices are noteworthy:

- **Grouped-Query Attention (GQA)**: Groups multiple attention heads to share keyvalue projections while keeping separate query projections, reducing memory usage without performance loss.
- **Rotary Position Embeddings (RoPE)**: Encodes relative positions directly into the query and key vectors, enabling better extrapolation to longer sequences.

RAG Backbone. The retrieval-augmented model is built on `meta-llama/Llama-3.2-1B-Instruct` without additional pretraining or fine-tuning. Greedy decoding is used with a maximum output length of 32 tokens.

3.3 Dataset and Retrieval Pipeline

Dataset Preparation. We follow the DPR NQ-Open split. The train and dev sets are downloaded as `.json.gz`, unpacked, and converted into Hugging Face `Dataset` objects. Each record contains a `question`, `answers`, and sets of positive, negative, and hard-negative contexts. Only the query text is given to the retriever during evaluation.

Retrieval. The BM25 retriever from Pyserini is used with the prebuilt `wikipedia-dpr-100w` index. The `search` function queries the index in parallel (up to 8 threads) to return the top-5 passages, each with `Title` and `Passage` fields.

3.4 Prompt Construction and Generation

Prompt Formatting. `make_augmented_inputs_for_generate` concatenates the question with the top-5 retrieved passages and then wraps them with a variant-specific prompt. For the Naive/CoT variants, each passage is rendered as a two-line block (title on the first line, passage text on the next), optionally prefixed with `Title:` and `Passage:`. For the Chat-style variant, we follow the exact code template:

```
<|start_header_id|>user<|end_header_id|>
{question}
{context}
<|start_header_id|>assistant<|end_header_id|>
The answer is:
```

Here, `{context}` is the raw concatenation of the five retrieved passages (title as the first line of each passage, followed by its body), without eliding the middle passages. (Note: any ellipses shown in figures are for presentation only and are not used in the actual model inputs.)

Generation. The `make_augmented_inputs_for_generate` function tokenizes the prompt, generates answers greedily, and applies variant-specific post-processing (reasoning in CoT, verification in Self-RAG, etc.).

3.5 Evaluation Metrics

Metrics. We report:

- **Exact Match (EM)**: The proportion of predictions exactly matching a gold answer; equivalent to classification accuracy.
- **ROUGE-1/2/L/Lsum**: Computed with `rouge_score`, measuring unigram, bigram, and longest common subsequence overlaps. ROUGE-Lsum averages LCS at the sentence level.

4 Results

4.1 GPT-small Benchmark Results

Before evaluating RAG configurations, we report the benchmark performance of the GPT-SMALL model across pretraining, classification, and summarization tasks. Table 1 shows the final loss and perplexity (PPL) where available.

Task	Final Step	Loss	PPL
Pretraining	93,844	3.0000	20.375
Classification	–	1.9354	–
Summarization	107,670	3.9594	–

Table 1: Benchmark results of GPT-SMALL across pretraining, classification, and summarization tasks.

4.2 RAG Performance by Method

Table 2 reports the performance of all evaluated RAG configurations on the NQ-Open DPR dataset, measured by Exact Match Accuracy (EM) and ROUGE metrics. The best score in each column is highlighted in bold.

Method	EM	R1	R2	RL	RLs
Naive RAG	0.1773	0.1538	0.0659	0.1527	0.1530
CoT Prompting	0.1578	**0.1782**	**0.0903**	**0.1773**	**0.1775**
Input Formatting	**0.1977**	0.1705	0.0790	0.1696	0.1700
Parsing Gen. Results	0.1446	0.1370	0.0615	0.1359	0.1362
Self-RAG	**0.1977**	0.1705	0.0790	0.1696	0.1700

Table 2: Main results on NQ-Open DPR. R1/2/L/Ls denote ROUGE-1/2/L/Lsum.

4.3 Observations

Naive RAG vs. CoT Prompting. Chain-of-Thought (CoT) prompting increases all ROUGE metrics substantially compared to Naive RAG (e.g., R-1: $+0.0244$, R-2: $+0.0244$, R-L: $+0.0246$), indicating that step-by-step reasoning encourages the model to produce richer, more contextually grounded answers. However, EM decreases by -0.0195, suggesting that the additional reasoning tokens may introduce paraphrasing or extraneous details that deviate from the exact gold answer span. This trade-off highlights that CoT may be more beneficial for metrics tolerant to lexical variation, while potentially harming strict exact-match metrics.

Impact of Input Formatting. Adopting the LLaMA-3.2-style `user–assistant` conversational prompt yields the highest EM (0.1977, $+0.0204$ over Naive RAG) and also improves ROUGE-L ($+0.0169$). This suggests that aligning the input format with the instruction-tuning distribution of the model can significantly improve factual precision without sacrificing recall-oriented metrics. Qualitatively, outputs generated under this format were more concise, with fewer incomplete sentences, indicating better adherence to the query focus.

Self-RAG. Combining Input Formatting with a prompt-based self-verification step produces identical scores to Input Formatting alone. We attribute this to two main factors: (1) the relatively small size of the backbone model (1B parameters), which limits its capacity to critically re-evaluate and revise answers, and (2) the use of greedy decoding (temperature $= 0$), which leads to deterministic repetition of the same answer even after the verification phase. Although no performance gain is observed here, the method retains strong potential: larger models, stochastic decoding strategies (e.g., nucleus sampling), and more targeted verification prompts could enable meaningful improvements.

5 Analysis and Discussion

Effectiveness of Prompt-Based Strategies. Our experiments demonstrate that prompt-based strategies can substantially affect zero-shot RAG performance. Chain-of-Thought prompting improved ROUGE scores by encouraging the model to produce richer, more descriptive answers. However, this came at the cost of lower EM due to deviations from exact reference strings. This trade-off highlights that prompt styles promoting reasoning can boost informativeness but may harm exact-match accuracy in fact-centric tasks.

Impact of Input Formatting. The user–assistant format aligned particularly well with the instruction-tuned LLaMA-3.2-1B-Instruct backbone, yielding the highest EM and competitive ROUGE scores. This suggests that matching the input structure to the model's pretraining and instruction-tuning style can be as impactful as more elaborate reasoning prompts.

Limitations of Self-Verification in Small Models. Self-RAG, which combines input formatting with a self-verification step, did not yield measurable gains over input formatting alone. We attribute this to the small model size (1B parameters) and the use of greedy decoding, which limits diversity between the initial and verified outputs. These limitations suggest that self-verification may require larger models and sampling-based decoding to realize its potential.

Retrieval Quality as a Bottleneck. Since the same top-5 BM25 passages were used across all variants, performance differences stem solely from prompt design and generation behavior.

While this isolates the effect of prompting, it also implies that improvements may be capped by retrieval quality. Integrating neural retrievers or hybrid retrieval could amplify the benefits of the proposed prompt strategies.

Future Directions. The findings indicate several promising extensions: (1) applying the strategies to larger instruction-tuned models, (2) exploring temperature-controlled or diverse decoding to enhance self-verification, and (3) combining prompt-based strategies with retrieval improvements. Such extensions may unlock greater gains in both factual accuracy and answer richness.

6 Conclusion

In this work, we investigated prompt-based strategies to improve zero-shot Retrieval-Augmented Generation (RAG) without any task-specific fine-tuning. We implemented a baseline Naive RAG pipeline and explored variants including Chain-of-Thought prompting, user–assistant input formatting, and a prompt-based self-verification method (Self-RAG). Our experiments on the NQ-Open DPR dataset show that: (1) reasoning-oriented prompts such as Chain-of-Thought can improve informativeness but may reduce exact-match accuracy, (2) aligning prompt structure with the models instruction-tuning style yields the largest EM gains, and (3) self-verification offers no immediate benefit for small models with greedy decoding, though it holds promise for larger models and more diverse generation settings.

These findings highlight that careful prompt design can produce measurable gains in zero-shot RAG, even without retrieval changes or model fine-tuning. Future work will explore combining prompt-based strategies with improved retrievers, scaling to larger models, and incorporating sampling-based decoding to fully realize the potential of self-verification.

The source code for this work is publicly available at https://github.com/snagjincha/RAG.

7 Related Work

Retrieval-Augmented Generation (RAG).
Lewis et al. (2020) introduced RAG, which couples a dense retriever (DPR; Karpukhin et al., 2020) with a seq2seq generator (BART) so that the decoder conditions on a small set of passages retrieved from a large corpus. The paper presents two variants, RAG-Sequence and RAG-Token, which differ in how retrieved passages are marginalised during generation. Follow-up retrieval-augmented approaches optimise other parts of the pipeline: REALM (Guu et al., 2020) jointly learns a retriever during LM pretraining by retrieving evidence for masked spans, and RETRO (Borgeaud et al., 2022) augments a decoder-only transformer with nearest-neighbour chunks retrieved at generation time from a large text database.

Prompt-based Adaptation.
Prompting enables task transfer without parameter updates. Brown et al. (2020) showed that GPT-3 can learn from demonstrations in the prompt; Li & Liang (2021) and Lester et al. (2021) proposed prefix/prompt tuning, attaching trainable embeddings to the input. Instruction tuning (Sanh et al., 2022; Chung et al., 2022) further improves zero-shot generalisation by supervising models on many task instructions.

Chain-of-Thought Prompting.
Wei et al. (2022) demonstrated that adding intermediate reasoning steps markedly improves multi-step tasks. Kojima et al. (2022) showed that even a single line ("Let's think step by step.") induces zero-shot CoT reasoning.

Self-verification and Answer Refinement.
Self-consistency (Wang et al., 2022) aggregates multiple reasoning paths to improve reliability. STaR (Zelikman et al., 2022) trains a critic to filter incorrect CoT traces, whereas Self-Refine (Madaan et al., 2023) lets an LLM iteratively critique and rewrite its own outputs.

Prompt-centric Improvements for RAG.
Shuster et al. (2021, R2-D2) and Lazaridou et al. (2022) added iterative rewrite loops to retrieval-based dialogue agents, yet systematic prompt-level self-verification inside the RAG pipeline remains under-explored. Our work closes this gap by combining (i) dialog-style prompt formatting, (ii) CoT reasoning, and (iii) a lightweight self-verification stage, yielding measurable gains in zero-shot RAG *without* any fine-tuning.

8 References

References

[1] ewis, Patrick, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Pavel Kuksa, Sewon Min, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, 2020.

[2] ei, Jason, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022.

[3] ouvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[4] arpukhin, Vladimir, Barlas Oguz, Sewon Min, Ledell Wu, Sergey Edunov, Danqi Chen, Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

[5] wiatkowski, Tom, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. In *Transactions of the Association for Computational Linguistics*, 2019.

[6] obertson, Stephen, Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. In *Foundations and Trends in Information Retrieval*, 2009.

[7] in, Chin-Yew. ROUGE: A package for automatic evaluation of summaries. In *Workshop on Text Summarization Branches Out (WAS)*, 2004.

[8] rown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

[9] i, Xiang Lisa, Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

[10] ester, Brian, Rami Al-Rfou, Noah Constant. The power of scale: Parameter-efficient adaptation for pretrained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021.

[11] anh, Victor, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, et al. Multitask prompted training enables zero-shot task generalization. In *International Conference on Learning Representations (ICLR)*, 2022.

[12] hung, Hyung Won, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

[13] ojima, Takeshi, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, Yusuke Iwasawa. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, 2022.

[14] ang, Xuezhi, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, et al. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

[15] elikman, Eric, Yuhuai Tony Wu, Noah D. Goodman, Yuling Chen. STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*, 2022.

[16] adaan, Aman, Shuyan Zhou, Uri Alon, Yiming Yang, Graham Neubig. Self-Refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, 2023.

[17] huster, Kurt, Spencer Poff, Moya Chen, Douwe Kiela, Jason Weston. Retrieval-augmented generation for knowledge-intensive NLP tasks. *arXiv preprint arXiv:2101.00117*, 2021.

[18] azaridou, Angeliki, Adhiguna Kuncoro, Elena Gribovskaya, Devang Agrawal, et al. Internet-augmented language models through few-shot prompting for open-domain question answering. *arXiv preprint arXiv:2203.05115*, 2022.

Appendix A: Experimental Environment

We ran all experiments on a single NVIDIA A100 GPU (80GB memory) with Python 3.11.13, PyTorch 2.5.1, and Pyserini 1.0.0.