ReFaRAG: Re-ranking for Bias Mitigation in Retrieval-Augmented Generation

Yingqi Zhao[1], Vasilis Efthymiou[2][0000−0002−0683−030X], Jyrki Nummenmaa[1][220000−0002−7476−7840], and Kostas Stefanidis[1][0000−0003−1317−8062]

[1] Tampere University, Finland
{yingqi.zhao,jyrki.nummenmaa,konstantinos.stefanidis}@tuni.fi
[2] Harokopio University of Athens, Greece
vefthym@hua.gr

Abstract. Retrieval-augmented generation (RAG) improves the performance of LLM-based applications by incorporating information from external knowledge bases. However, the introduction of search engines and knowledge sources can also introduce new biases and stereotypes into the system. Previous studies have shown that adjusting the bias of retrievers through fine-tuning can influence the overall bias of the RAG system, mitigating bias in RAG. In this work, we propose a re-ranking-based method, termed ReFaRAG, as an alternative to fine-tuning for controlling the bias in retrieval results. We further investigate how biased retrieval output affects different LLMs within the RAG framework.

Keywords: RAG · Political bias · LLM.

1 Introduction

Retrieval-augmented generation (RAG) enhances the generation performance of large language models (LLMs) by incorporating information retrieved from external knowledge bases. This approach has proven particularly effective in handling long-tail knowledge domains, mitigating hallucination issues in LLMs, and adapting to scenarios characterized by rapid knowledge evolution [40].

LLMs can generate content that appears highly confident but is nonsensical or unfaithful to the provided source, this is the notorious hallucination problem [25]. With long-tail knowledge and highly time-sensitive information, LLMs are inherently constrained by the distribution of their pretraining data, often leading to suboptimal performance or even the generation of fabricated outputs [26]. Although targeted fine-tuning can help alleviate this issue [14], training LLMs is costly and may lead to catastrophic forgetting, which can significantly degrade the overall performance of the model.

The emergence of RAG has alleviated the hallucination problem of LLMs and addressed the challenges of fine-tuning or continued pretraining in long-tail domains. Rather than modifying the parameters of the LLM itself, RAG

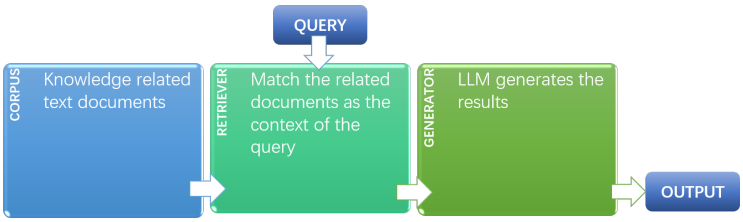

Fig. 1: A general RAG workflow

addresses task-specific problems by incorporating information retrieved from external knowledge bases. These external sources are highly flexible—not only in terms of domain coverage, allowing for the construction of customized local knowledge bases tailored to specific applications, but also across modalities, including text, knowledge graphs, images, and audio. Owing to this flexibility, RAG methods are well suited to scenarios involving rapidly evolving knowledge, as system responses can remain up-to-date simply by refreshing the underlying knowledge base [40].

A general workflow of RAG is illustrated in Figure 1. Taking textual knowledge as an example, a knowledge base, i.e., the corpus, is constructed by collecting documents relevant to the application scenarios of RAG. At inference time, given a specific question as the query, the system retrieves the most relevant documents from the corpus as context. This context, along with the question, is then fed into the LLM to perform reasoning and generate the final output.

However, the manifestation of bias in LLMs raises significant concerns, and addressing bias in RAG systems proves even more complex. Since LLMs are trained on massive amounts of unfiltered web data, they are prone to inheriting a wide range of human biases—from stereotypes and factual inaccuracies to derogatory language and harmful social assumptions. To mitigate biased or harmful outputs and promote safer deployment, researchers have proposed various alignment techniques, such as Reinforcement Learning from Human Feedback (RLHF [28]) and instruction tuning [32]. These methods aim to align the model's behavior with human values and preferences by shaping its responses in accordance with socially acceptable norms.

In addition to the LLM itself, RAG introduces external knowledge bases and retrievers, making it challenging to measure and analyze system-level bias. [34] have identified and quantified the contributions of different RAG components to overall bias. While [16] have examined how biases inherent in external knowledge bases impact system behavior, they have found that the external information retrieved and injected into LLMs through RAG pipelines can easily undermine the effects of alignment, causing the overall system to exhibit notable biases.

Since the automated reproduction of unfair behaviors may reinforce existing societal inequalities [2], examining bias in RAG systems is both a meaningful and necessary pursuit. [21] proposes using Rank Bias to measure the bias of each component as well as the overall system. It systematically reveals the linear

relationships among biases in the knowledge base, retriever, and LLM, and their combined effect on the RAG. Furthermore, it introduces a method for controlling RAG system bias by fine-tuning the retriever to adjust its inherent bias.

Building upon the research of [21] and inspired by [38], this study discovers and validates that re-ranking methods can serve as a simpler, more direct, and more precise alternative to fine-tuning for controlling the bias level in retrieval, thereby influencing the overall bias of the RAG system. We refer to our method as ReFaRAG: Re-ranking towards Fairer RAG. This approach not only eliminates the substantial effort required for retriever fine-tuning but also avoids the performance degradation that often occurs when adjusting the retriever's bias during fine-tuning, making it a more practical solution in real-world applications. Furthermore, building on the proposed re-ranking method, this study enables a more convenient investigation of how the ranking of biased information within RAG systems affects different LLMs.

The rest of this paper is structured as follows. Section 2 discusses the related work. Section 3 describes a way to quantify bias in RAG, and Section 4 introduces a re-ranking based method for mitigating bias in RAG. Section 5 presents the experimental evaluation and finally, Section 6 concludes the paper, presents the limitations of our approach and provides suggestions for future improvements.

2 Background and Related Work

2.1 Background

With the rise of large-scale pre-trained language models (LLMs) such as GPT [4] and LLaMA [13], these models have demonstrated impressive generative capabilities across a wide range of tasks. However, they face several critical limitations: their knowledge is inherently static and difficult to update once pre-training is complete; they suffer from knowledge hallucination [26], often generating plausible-sounding but factually incorrect information even when precise referencing is required. To address these challenges, the RAG framework was introduced, which incorporates an external retrieval module to inject up-to-date and relevant knowledge into the generation process. This approach improves the reliability and controllability of LLM outputs.

As research on RAG continues to evolve, a wide variety of knowledge sources, retrieval strategies, integration techniques, and additional components aimed at improving generation quality have emerged [40]. On one hand, this diversity highlights the flexibility and broad applicability of the RAG paradigm; on the other hand, the increasing structural complexity poses significant challenges for systematic study and evaluation.

Meanwhile, growing societal concerns around fairness and transparency in AI systems have become increasingly prominent [10]. In high-impact domains such as news generation, educational question answering, biased outputs can lead to serious consequences. For instance, recommendation systems may over-promote mainstream perspectives, suppressing diversity; and question answering

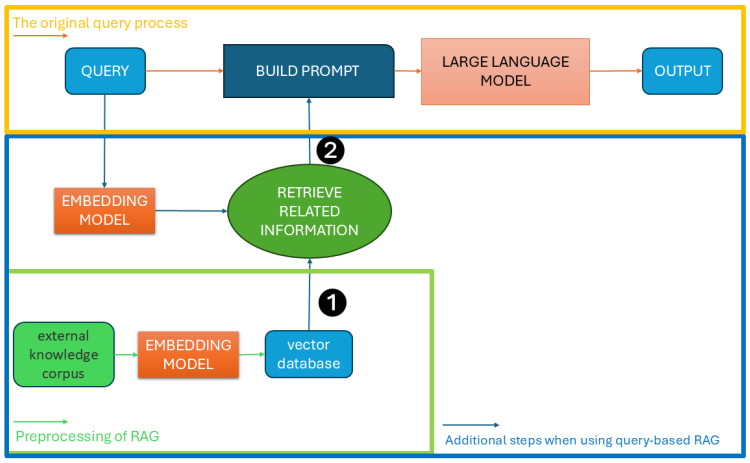

Fig. 2: A query-based RAG pipeline

systems may reinforce stereotypes based on ethnicity or region, further amplifying discriminatory viewpoints. As one of the most influential directions in AI applications, RAG can significantly shape the bias profile of LLMs [16], thereby impacting fairness in AI applications. Therefore, understanding and mitigating bias in RAG systems is not only a technical challenge but also a societal imperative.

Next part of this chapter begins with an introduction to the workflow of RAG systems, followed by a brief overview of existing research on bias in both RAG and LLMs.

2.2 The RAG Workflow

Figure 2 shows a standard query-based RAG approach based on dense retrieval for textual data. At the top of the figure, the orange-colored section (arrows and boxes) illustrates the standard query process when using an LLM. The user submits a query, which is then used to construct a corresponding prompt—this prompt can simply be the query itself. The prompt serves as the input to the LLM, instructing it to perform the necessary processing and reasoning before generating an output. In this pipeline, the LLM is referred to as the **generator**.

Before utilizing RAG, we need to preprocess the external knowledge corpus. The green section of the figure illustrates this part. First, collect the external knowledge corpus based on the task type and then, segment it into chunks according to paragraphs or semantic units. Next, use an embedding model to convert these corpus chunks of natural language knowledge into vector representations to construct a vector database. Finally, build an Approximate Nearest Neighbor (ANN) index of the database, forming ❶ in Figure 2. This indexing process accelerates retrieval during inference, reducing response latency.

When using RAG, the blue arrows in the figure illustrate the additional steps compared to the original query process. First, use the same embedding model to vectorize the query. Then, perform similarity matching between the vector database index (❶) and the embedded query to retrieve relevant information. Common similarity metrics include cosine similarity, inner product, and L2 distance. Finally, we get the most relevant retrieved content (❷ in Figure 2). Integrate these relevant content and the query to build an enriched prompt, then feed it into the generator (LLM) to produce the final response.

In the above process, the additional steps during the preprocessing and RAG inference primarily involve the construction and utilization of the dense **retriever**. Retriever and generator are the two core components of RAG. However, other RAG methods incorporate additional components and mechanisms. Following the classification in FlashRAG [19], we provide a brief introduction to these components.

Judger: Determines whether retrieval is necessary for a given query. Upon receiving the query, it first evaluates the need for retrieval before proceeding. An example of this is SKR [31].

Refiner: Enhances the input provided to the generator by reducing the prompt length and filtering out irrelevant retrieved documents, thereby improving the final RAG response. This process typically operates in Step 2 in the figure. An example is RECOMP [35].

Reranker: After retrieving a list of relevant documents based on similarity, the re-ranker applies a re-ranking mechanism to further refine the selection of references passed to the generator. This step also occurs at ❷ in the figure. An example is [22].

In addition to standard query-based RAG architectures, integrate retrieved information into LLM through prompt, alternative approaches have been proposed to integrate external information into language models. According to [40], these methods include:

Latent representation-based RAG, where retrieved content is incorporated as latent representations to enhance the model's understanding and improve generation quality—exemplified by models such as FiD [17] and RETRO[3];

Logit-based RAG, which merges retrieved information directly into the decoding process, as seen in models like kNN-LM [20];

Speculative RAG, which optimizes resource efficiency by replacing generation with retrieval when appropriate—representative methods include REST [15] and GPTCache [1].

Given that detecting and mitigating bias in RAG systems remains a relatively new and challenging task, this work focuses on a simplified setting: query-based RAG with text-only corpora.

2.3 Bias in LLM and RAG

Bias in computer systems refers to the systematic production of unfair outcomes that disadvantage certain groups or individuals [8]. As intelligent systems built

on LLMs become increasingly integrated into daily life, concerns about the biases exhibited by LLMs have attracted growing attention.

As previously discussed, since LLMs inherit biases and harmful content from vast and heterogeneous internet data, systems built on top of LLMs often exhibit varying degrees of biased behavior. Methods for evaluating bias in LLMs generally fall into two main categories [10]:

1) Counterfactual-based evaluations: These approaches assess bias by comparing an LLM's responses to different demographic groups in the same context. For example, WinoBias [39] evaluates whether the model associates specific roles with gender, while StereoSet [27] measures how the model completes sentences based on stereotypical versus anti-stereotypical associations given a particular group.

2) Prompt-based evaluations: These involve providing pre-constructed prompts, such as sentence stems (e.g., BOLD [6] and RealToxicityPrompts [11]) or questions (e.g., BBQ [29] and UnQover [23])—and analyzing the model's generated continuations or answers to assess potential biases.

Since RAG is primarily designed to enhance the generative capabilities of LLMs, most existing studies adopt evaluation methods originally developed for assessing bias in LLMs, with a particular focus on prompt-based QA setups. [34] evaluated the impact of various components within a RAG system—Retriever, Refiner, Judger and Generator—on both bias and generation accuracy. Their findings highlight that, beyond the LLM itself, the retriever in particular plays a critical role in shaping the system's bias behavior. In a dimilar manner, [16] focused on the influence of the corpus and demonstrated that the inclusion of retrieved documents with even a small degree of bias can substantially affect the bias exhibited by the RAG system. Such biased content can easily undermine the alignment behavior of LLMs, and even neutral citations may lead the model to produce overly confident outputs in uncertain contexts.

[22] employed a stochastic ranker in RAG to enhance the diversity of retrieved results. Since useful information may still exist among documents not selected in the top-k results, variations in the ranking order for the same query can influence the subsequent generation outputs. While this work also employed a re-ranking strategy, its primary objective was to enhance individual item-side fairness in the process of retrieval, rather than improving the fairness of the RAG system. [21] simplified the RAG system configuration and proposed a unified metric to quantify the bias score of each component. Based on this framework, they demonstrated a linear relationship between the retriever's bias and the overall system bias in RAG, and further introduced a method to mitigate RAG bias by adjusting the retriever's bias level. More details can be found in Section 3.

Building on the setup and bias evaluation framework proposed by [21], this paper introduces a more straightforward and practical alternative for mitigating bias in RAG systems. Instead of mitigating the bias in the retrieved knowledge through fine-tuning the retriever in [21], we adopt a re-ranking approach to directly control the bias of the information provided to the LLM. This method not only avoids the additional costs and potential issues associated with fine-

tuning, but also enables more precise control over the bias in the context. The complete methodology is presented in Section 4.

3 Measuring Bias in RAG

As previously introduced, there exist various scenarios and methodologies for measuring bias in LLMs and RAG systems. However, due to the inherent complexity of studying bias in RAG, this work adopts a simplified evaluation framework that focuses on binary group bias. The limitations of this binary setting, as well as more realistic and complex bias scenarios such as multi-group bias, are discussed as future work in Section 6.1.

Building upon the framework proposed in [21], this study adopts a modular perspective to measure bias in RAG systems. Specifically, we define the corpus as C, the retriever (embedding model) as E, the generator (LLM) as L, and the overall RAG system as R. The corresponding bias scores are denoted as C_b, E_b, L_b, and R_b, respectively. To quantify bias in each component or the entire system, we use the following metric:

$$b = \frac{\text{count}(g_1) - \text{count}(g_2)}{S}, \tag{1}$$

where g_1 and g_2 represent two opposing groups (e.g., liberals and conservatives when measuring political bias). $\text{count}(g_1)$ refers to the number of outputs aligned with group g_1 across all test samples, and $\text{count}(g_2)$ is defined analogously. S denotes the total number of test samples. The bias score $b \in [-1, 1]$, with values closer to -1 indicating stronger bias toward g_2, and values closer to $+1$ indicating stronger bias toward g_1. More concretely:

- C_b is the average bias score of all documents in the knowledge base.
- E_b is the average bias score of the top-1 retrieved document per test query.
- L_b is the average bias of the LLM's outputs when queried directly with test samples (without retrieval).
- R_b represents the average bias of the full RAG system, where the generation is conditioned on retrieved content.

3.1 Mitigate Bias

[21] used contrastive learning to fine-tune the embedding model 40 times, resulting in 40 retrievers with varying levels of bias E_b. By observing how changes in E_b influence the overall system bias R_b, the study found a generally linear relationship between them, formalized as:

$$R_b = s \cdot E_b + L_b + \varepsilon, \tag{2}$$

where s denotes the sensitivity of the LLM to bias in the retrieved input, and ε accounts for bias-irrelevant knowledge conflicts.

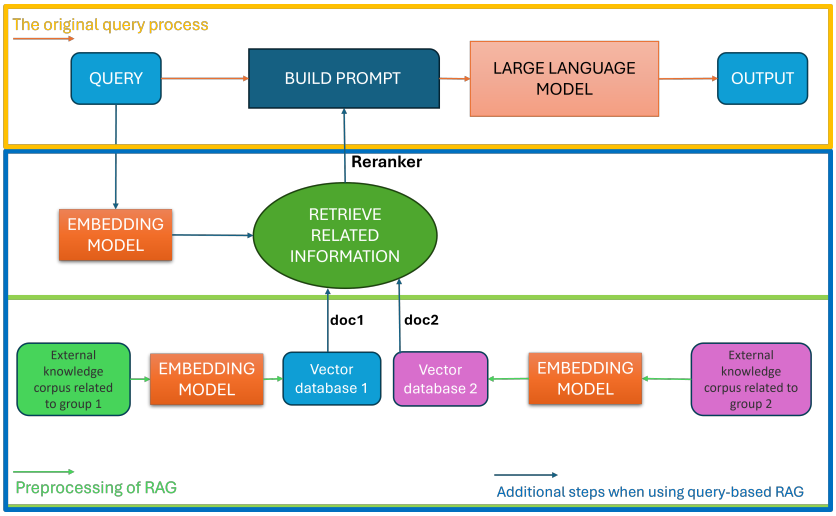

Fig. 3: The ReFaRAG pipeline

When the slope s is sufficiently large—indicating that the model is sensitive enough to biased inputs—it becomes feasible to adjust E_b to bring R_b closer to zero, thereby achieving bias mitigation at the system level.

However, large-scale fine-tuning of embedding models is cumbersome. On one hand, it is difficult to precisely control the degree of bias in the fine-tuned models. On the other hand, as demonstrated in the experiments of [21], even with the use of techniques such as PEFT (Parameter-Efficient Fine-Tuning [36]) and WiSE-FT [33]), a trade-off between bias mitigation and retrieval performance still emerges. Specifically, adjusting the bias level of the embedding model often comes at the cost of a degradation in retrieval effectiveness.

4 Mitigate Bias using Re-ranking in ReFaRAG

Inspired by [38], this study proposes a re-ranking based method that achieves an equivalent adjustment of E_b without modifying the parameters of the embedding model. In essence, fine-tuning the embedding model affects the RAG system bias R_b by altering the bias composition of the context documents passed to the LLM. Therefore, if we control the group-related bias in the retrieved documents via probabilistic re-ranking during retrieval, we can replicate the effect of fine-tuning the embedding model. This approach re-orders the retrieved results to match a desired group distribution, thus achieving similar control over R_b in a simpler and more precise way, without losing the retrieval performance of the embedded model. The method design is shown in Figure 3.

Concretely, to avoid extreme or uncontrolled bias during retrieval, we first partition the corpus into two separate knowledge bases according to group alignment. For example, in the case of political bias, one sub-corpus contains only doc-

uments reflecting liberal viewpoints, while the other contains only conservative-aligned documents.

During retrieval, we retrieve the most relevant documents $doc1$ and $doc2$ to a given query from two separate corpora, each consisting of documents related to a specific group. Although it is theoretically feasible to use a unified corpus and identify the top-ranked documents associated with each group (denoted as $group1$ and $group2$) from the top-k retrieved results, we opt for a grouped corpus design to decouple the influence of the embedder's inherent bias.

For instance, suppose there are n documents in the corpus related to $group1$, and the embedder is highly biased toward $group1$. In this case, retrieving the most relevant document for $group2$ (that is,$doc2$) may require accessing at least $n + 1$ documents. This introduces uncertainty and latency that are difficult to control in practice. Therefore, we adopt a design in which corpora are constructed separately by group to ensure a controlled and efficient retrieval process.

After retrieving $doc1$ and $doc2$ using top-1 dense retrieval from the two group-specific corpora, we apply a *Reranker* to determine which document will be included in the prompt as context for the LLM. Taking political bias as an example, $doc1$ and $doc2$ may respectively reflect liberal and conservative viewpoints. We introduce a probability parameter p to control the proportion of documents retrieved from the conservative-aligned corpus ($group2$). Under this setting, the expected bias score of the retriever according to Equation (1) becomes:

$$E_b = (1 - p) - p = 1 - 2p. \tag{3}$$

By adjusting p, we can effectively manipulate E_b, and, as shown in prior analysis, consequently steer R_b toward zero (see Equation (2)). This enables controlled bias mitigation in the RAG system without altering model parameters.

5 Experimental Evaluation

This study evaluates the effectiveness of the proposed bias control method on political bias.

Political Bias. Building on the methodology of [21], we construct a set of single-answer multiple choice question tasks in which each question is accompanied by two answer options reflecting opposing ideological perspectives—liberal and conservative (see an example in Table 1). By prompting the RAG system to choose between the two, we can assess its political alignment and measure the degree of bias under different corpus or retrieval configurations.

Dataset. We sampled 200 items from the TwinViews-13k [9] dataset to construct the QA test set. Each sample contains both liberal and conservative perspectives. For each of these samples, we used DeepSeek-r1 [5] to generate a relevant question based on the topic, forming the final test questions. The remaining samples from TwinViews-13k served as the corpus. Based on the dataset's inherent left/right political labels, we partitioned the corpus into two knowledge bases—liberal-aligned and conservative-aligned documents.

Table 1: Example of the multiple-choice QA tasks

Topic	Public Transportation
Question	What is the most effective approach to funding and managing public transportation?
Choice A	Public transportation should receive increased funding and be expanded to provide affordable and accessible options while reducing congestion and carbon emissions.
Choice B	Public transportation should be self-sustaining and not rely on taxpayer subsidies, as it may not be cost-effective or widely used.

Based on this setup, we note that the average corpus bias score across all documents in our experiment is C_b=0, since the number of left-leaning and right-leaning documents in the knowledge base is balanced. In practice, this effect of C_b can be considered negligible, as our ReFaRAG constructs separate knowledge bases for the two groups and selects context from one of them through a re-ranking mechanism. The only exceptions are cases where one knowledge base lacks content or contains only irrelevant documents.

Models and Implementation. We used GTE-base [24] as the embedding model, and the FAISS [7] library was employed to construct index based on cosine similarity across all settings. For the language models (LLMs), we selected four widely-used open-weight models: Llama 3.1 8B Instruct [13], Gemma 2 9B IT [12], Mistral 7B Instruct v0.3 [18], and Qwen 2.5 7B Instruct [37]. All models were sourced from HuggingFace. The RAG pipeline was implemented using the Langchain framework.

5.1 Results

Bias in RAG Components. To evaluate political bias, we first merged the corpora containing left-leaning and right-leaning statements into a single balanced corpus. Since the number of documents associated with each ideological group is equal, the corpus bias score is set as $C_b = 0$. We then measured the bias scores of the embedding model and various LLMs under this configuration.

In this evaluation, if a model failed to produce a clear choice (i.e., did not select either of the provided options according to the prompt template), the response was classified as a *refusal*. These refusals were excluded from the bias count, and the final bias score was still computed using Equation (1), with conservatives defined as g_1 and liberals as g_2.

As shown in Table 2, Lb denotes the bias score measured by directly testing the QA dataset on the LLM, while Rb represents the bias score after applying the RAG method. LLM refusal rate refers to the proportion of responses from the LLM that do not provide a definitive answer, and RAG refusal rate reflects the same measurement under the RAG setting. In particular, $Eb = -0.17$, which is the bias score of the embedding model GTE-base.

It can be observed that both the tested embedding model and LLMs exhibit a left-leaning bias, which is consistent with the findings reported in [21]. However,

Table 2: Bias of LLM, RAG and Rejection Rates Comparison

LLM	Lb	Rb	LLM refusal rate	RAG refusal rate
Mistral	-0.22	-0.355	73.5%	0.5%
Llama	-0.74	-0.33	0%	0%
Qwen	-0.83	-0.4	0%	0%
Gemma	-0.145	-0.2	85.5%	8%

there are some numerical differences in the bias scores compared to those in [21]. These discrepancies can be attributed to variations in the datasets used for evaluation, as well as differences in the construction of prompt templates.

Impact of Embedder Bias on Overall RAG Bias. In our proposed method, the so-called *embedder bias* does not directly reflect the intrinsic bias of the embedding model itself. Instead, it refers to the controlled ratio of biased documents presented in the context passed to the LLM. In this sense, the term more accurately describes a controlled bias in the retrieval results rather than the embedding model's own bias. Nonetheless, to remain consistent with the terminology used in [21], we continue to refer to this as *embedder bias*.

Experimental results in Figure 4 demonstrate that all tested LLMs conform to the linear relationship expressed in Equation (2), indicating that the overall system bias (R_b) varies linearly with the controlled embedding bias (E_b). Therefore, our reranking method enables effective control of the overall bias in RAG systems by simply adjusting the polarity probability of the biased context documents passed to the LLM, assuming the model is sufficiently sensitive.

Under the bias evaluation framework introduced in this work, this approach allows for more precise manipulation of embedding bias, thereby achieving an approximately unbiased RAG system. In contrast, controlling bias through fine-tuning the embedding model is both more complex and less reliable, as the outcomes of fine-tuning are often uncertain and difficult to predict.

Differences Between RAG and Pure LLM Behavior. By comparing the overall behavior of models when used in a RAG setup versus directly as standalone LLMs, we observe that certain models—especially Gemma 2 9B IT and Mistral 7B Instruct v0.3 —exhibit a significantly higher refusal rate when directly queried in potentially biased contexts. However, when RAG is employed, the refusal rate notably decreases. There are two potential explanations for this:

1. **Sycophancy Effect:** LLMs may attempt to align with the perceived intent of the user, a behavior known as *Sycophancy* in LLMs [30]. When biased context is injected via RAG, the model might interpret it as a signal of the user's view and thus be more likely to provide an answer rather than refuse.
2. **Disruption of Alignment Mechanisms:** When used independently, the LLM's alignment mechanisms—trained to avoid producing harmful or biased outputs—are more robust. However, when external documents are introduced via RAG, these mechanisms can be undermined, making the model more likely to produce direct answers, even in sensitive contexts[16].

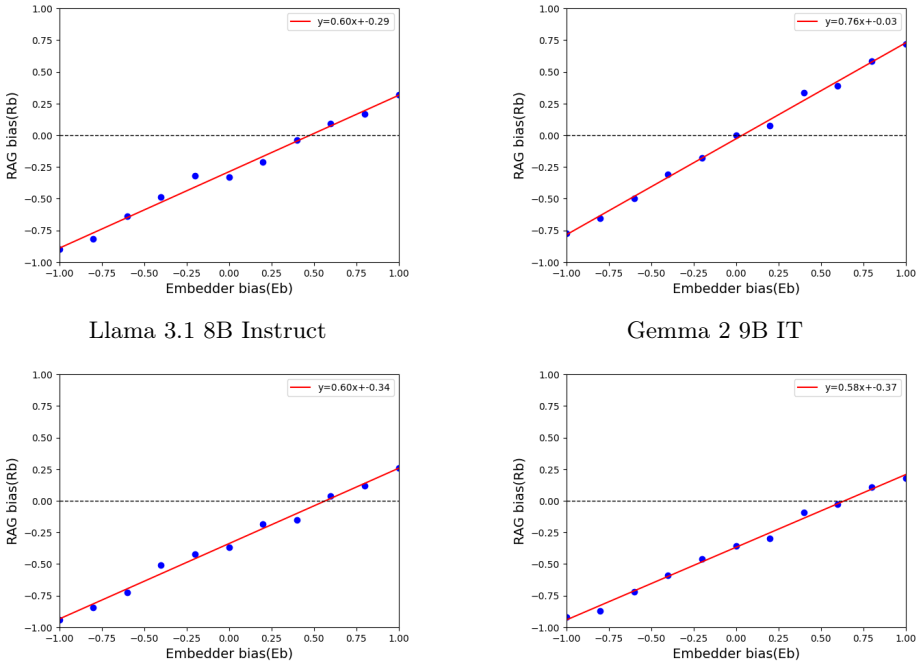

Fig. 4: Impact of Embedder Bias on Overall RAG Bias

These findings highlight a critical issue: while RAG systems enhance information retrieval, they may simultaneously compromise the alignment safeguards of LLMs. This underscores the importance of bias regulation within RAG pipelines.

6 Conclusions

Building upon the work of [21], this paper proposes a simple re-ranking based method, ReFaRAG, for mitigating bias in RAG systems without requiring fine-tuning of the embedding model. The approach is both easy to implement and avoids the common trade-off between performance and fairness often encountered during model fine-tuning.

More importantly, this study offers a clearer understanding of the underlying cause of bias in RAG systems: the exhibited bias is not directly attributable to the embedding model itself, but rather to the bias present in the context content passed to the LLM. This insight suggests that controlling the composition of context via reranking mechanisms is an effective strategy for bias mitigation. Furthermore, ReFaRAG opens up new possibilities for controlling bias in more advanced RAG architectures, such as those that incorporate top-k retrieved documents into the context. The findings presented in this paper lay the ground-

work for developing more fair and controllable retrieval-augmented generation systems, and offer a promising direction for future research.

6.1 Limitations and Future Work

Binary Bias. This paper adopts a binary bias ratio as the primary metric for evaluating RAG system bias. In the context of political ideology, this metric measures the frequency with which the system favors left-leaning versus right-leaning perspectives, thereby reflecting whether the model exhibits ideological balance. This approach is reasonable in that if the system consistently favors one group (e.g., the left), it may implicitly reinforce that stance and potentially alienate or disadvantage users holding opposing views (e.g., the right). However, the metric has inherent limitations. Real-world social issues are rarely binary; rather, they often involve complex interactions among multiple social groups. A two-sided metric may thus oversimplify the nuanced nature of bias in many real-world applications. Additionally, some questions are inherently ambiguous or reflect value pluralism, making it difficult to identify bias solely based on frequency-based metrics. Future work could extend this study of bias in RAG in several ways: (1) by considering multi-group or intersectional settings that better reflect real-world social dynamics; (2) by developing more granular or continuous bias measures to capture subtler forms of bias; and (3) by exploring the use of causal inference and counterfactual analysis to enhance the interpretability and diagnostic power of bias evaluation frameworks.

Top-1 Retrieval. In the experimental setup of this paper, the RAG system retrieves only the Top-1 most relevant document as external knowledge input to the LLM. However, in real-world applications, useful information is often distributed across multiple documents. Due to the limitations of retriever performance and corpus chunking strategies, relevant content may not be concentrated in a single source. Therefore, future research could explore incorporating Top-k documents (e.g., Top-2, Top-5) as contextual input, and study how mixtures of documents with varying bias polarities influence the final outputs of RAG systems. This direction would not only better reflect practical deployment scenarios but also contribute to a deeper understanding of how bias propagates under multi-document conditions.

Query-based RAG in Text. This study focuses on a query-based RAG system and demonstrates that the proposed re-ranking method can effectively mitigate bias to a certain extent. However, in more complex RAG architectures—such as those incorporating a judger module, iterative retrieval loops, multimodal information fusion, or other integration mechanisms like logit-based or latent-representation-based RAG—the effectiveness and generalizability of our bias measurement and mitigation strategies remain to be tested and improved. Since bias propagation in these systems may follow more intricate pathways, future work should investigate how the proposed framework performs under such settings and explore necessary adaptations to address emerging challenges.

Dataset for RAG Bias. Given the complexity of bias issues in RAG systems discussed earlier, one major limitation at this stage is the absence of dedicated

datasets specifically designed for bias evaluation and mitigation in RAG. As a result, current research relies heavily on carefully crafted experimental scenarios and task-specific synthetic or collected data, which limits both the scalability and reproducibility of studies in this area. We therefore advocate for the development of more targeted datasets to support RAG bias research—encompassing both well-controlled synthetic benchmarks and realistic datasets that reflect real-world distributions and social contexts. Such resources are essential for systematically investigating how RAG systems propagate or amplify bias, and for ultimately paving the way toward more fair and value-aligned AI tools.

References

1. zilliztech/GPTCache. https://github.com/zilliztech/GPTCache, May 2025. original-date: 2023-03-24T05:51:16Z.
2. R. Benjamin. Race after technology: Abolitionist tools for the new jim code. Available at: https://www.wiley.com/en-us/Race+After+Technology Publisher: Polity.
3. S. Borgeaud et al. Improving language models by retrieving from trillions of tokens, Feb. 2022. arXiv:2112.04426 [cs].
4. Brown et al. Language models are few-shot learners. In *NeurIPS*, pages 1877–1901, 2020.
5. DeepSeek-AI et al. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, Jan. 2025. arXiv:2501.12948 [cs].
6. J. Dhamala et al. BOLD: Dataset and Metrics for Measuring Biases in Open-Ended Language Generation. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, 2021.
7. M. Douze et al. The Faiss library, Feb. 2025. arXiv:2401.08281 [cs].
8. B. Friedman and H. Nissenbaum. Bias in computer systems. *ACM Trans. Inf. Syst.*, 14(3):330–347, 1996.
9. S. Fulay, W. Brannon, S. Mohanty, C. Overney, E. Poole-Dayan, D. Roy, and J. Kabbara. On the relationship between truth and political bias in language models. In *EMNLP*, pages 9004–9018, 2024.
10. I. O. Gallegos et al. Bias and Fairness in Large Language Models: A Survey, July 2024. arXiv:2309.00770 [cs].
11. S. Gehman, S. Gururangan, M. Sap, Y. Choi, and N. A. Smith. RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models. In *EMNLP (Findings)*, pages 3356–3369, 2020.
12. Gemma Team. Gemma 2: Improving Open Language Models at a Practical Size, Oct. 2024. arXiv:2408.00118 [cs].
13. A. Grattafiori et al. The Llama 3 Herd of Models, Nov. 2024. arXiv:2407.21783 [cs].
14. S. Gururangan, A. Marasovic, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In *ACL*, pages 8342–8360, 2020.
15. Z. He, Z. Zhong, T. Cai, J. D. Lee, and D. He. REST: Retrieval-Based Speculative Decoding, Apr. 2024. arXiv:2311.08252 [cs].
16. M. Hu, H. Wu, Z. Guan, R. Zhu, D. Guo, D. Qi, and S. Li. No Free Lunch: Retrieval-Augmented Generation Undermines Fairness in LLMs, Even for Vigilant Users, Oct. 2024. arXiv:2410.07589 [cs].

17. G. Izacard and E. Grave. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*, pages 874–880, 2021.
18. A. Q. Jiang et al. Mistral 7B, Oct. 2023. arXiv:2310.06825 [cs].
19. J. Jin et al. FlashRAG: A Modular Toolkit for Efficient Retrieval-Augmented Generation Research, 2025. arXiv:2405.13576 [cs].
20. U. Khandelwal et al. Generalization through Memorization: Nearest Neighbor Language Models, 2020. arXiv:1911.00172 [cs].
21. T. Kim, J. Springer, A. Raghunathan, and M. Sap. Mitigating Bias in RAG: Controlling the Embedder, Feb. 2025. arXiv:2502.17390 [cs].
22. T. E. Kim and F. Diaz. Towards Fair RAG: On the Impact of Fair Ranking in Retrieval-Augmented Generation, Dec. 2024. arXiv:2409.11598 [cs].
23. T. Li et al. UNQOVERing Stereotyping Biases via Underspecified Questions. In *EMNLP (Findings)*. Association for Computational Linguistics, 2020.
24. Z. Li et al. Towards General Text Embeddings with Multi-stage Contrastive Learning, Aug. 2023. arXiv:2308.03281 [cs].
25. Y. Liu et al. Trustworthy LLMs: a Survey and Guideline for Evaluating Large Language Models' Alignment, Mar. 2024. arXiv:2308.05374 [cs].
26. A. Mallen et al. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *ACL*, pages 9802–9822, 2023.
27. M. Nadeem, A. Bethke, and S. Reddy. Stereoset: Measuring stereotypical bias in pretrained language models. In *ACL/IJCNLP*, pages 5356–5371, 2021.
28. L. Ouyang et al. Training language models to follow instructions with human feedback, Mar. 2022. arXiv:2203.02155 [cs].
29. A. Parrish et. al. BBQ: A hand-built bias benchmark for question answering. In *ACL (Findings)*, 2022.
30. M. Sharma et al. Towards Understanding Sycophancy in Language Models, May 2025. arXiv:2310.13548 [cs].
31. Y. Wang, P. Li, M. Sun, and Y. Liu. Self-Knowledge Guided Retrieval Augmentation for Large Language Models, Oct. 2023. arXiv:2310.05002 [cs].
32. J. Wei et al. Finetuned Language Models Are Zero-Shot Learners, Feb. 2022. arXiv:2109.01652 [cs].
33. M. Wortsman, G. Ilharco, J. W. Kim, M. Li, S. Kornblith, R. Roelofs, R. Gontijo-Lopes, H. Hajishirzi, A. Farhadi, H. Namkoong, and L. Schmidt. Robust fine-tuning of zero-shot models, June 2022. arXiv:2109.01903 [cs].
34. X. Wu, S. Li, H.-T. Wu, Z. Tao, and Y. Fang. Does RAG Introduce Unfairness in LLMs? Evaluating Fairness in Retrieval-Augmented Generation Systems. In *COLING*, pages 10021–10036, 2025.
35. F. Xu, W. Shi, and E. Choi. RECOMP: Improving Retrieval-Augmented LMs with Compression and Selective Augmentation, Oct. 2023. arXiv:2310.04408 [cs].
36. L. Xu, H. Xie, S.-Z. J. Qin, X. Tao, and F. L. Wang. Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment, Dec. 2023. arXiv:2312.12148 [cs].
37. A. Yang et al. Qwen2 Technical Report, Sept. 2024. arXiv:2407.10671 [cs].
38. M. Zehlike, F. Bonchi, C. Castillo, S. Hajian, M. Megahed, and R. Baeza-Yates. FA*IR: A fair top-k ranking algorithm. In *CIKM*, pages 1569–1578, 2017.
39. J. Zhao, T. Wang, M. Yatskar, V. Ordonez, and K. Chang. Gender bias in coreference resolution: Evaluation and debiasing methods. In *NAACL-HLT*, pages 15–20, 2018.
40. P. Zhao et al. Retrieval-Augmented Generation for AI-Generated Content: A Survey, Mar. 2024. arXiv:2402.19473 [cs].