Retrieval Augmented Generation with Collaborative Filtering for Personalized Text Generation

Teng Shi
Jun Xu*
Xiao Zhang
Gaoling School of Artificial Intelligence
Renmin University of China
Beijing, China
{shiteng,junxu,zhangx89}@ruc.edu.cn

Xiaoxue Zang
Kai Zheng
Kuaishou Technology Co., Ltd.
Beijing, China
xxic666@126.com
zhengk92@gmail.com

Yang Song
Han Li
Kuaishou Technology Co., Ltd.
Beijing, China
ys@sonyis.me
lihan08@kuaishou.com

Abstract

Recently, the personalization of Large Language Models (LLMs) to generate content that aligns with individual user preferences has garnered widespread attention. Personalized Retrieval-Augmented Generation (RAG), which retrieves relevant documents from the user's history to reflect their preferences and enhance LLM generation, is one commonly used approach for personalization. However, existing personalized RAG methods do not consider that the histories of similar users can also assist in personalized generation for the current user, meaning that collaborative information between users can also benefit personalized generation. Inspired by the application of collaborative filtering in recommender systems, we propose a method called **CFRAG**, which adapts **C**ollaborative **F**iltering to **RAG** for personalized text generation. However, this presents two challenges: (1) how to incorporate collaborative information without explicit user similarity labels? (2) how to retrieve documents that support personalized LLM generation? For Challenge 1, we use contrastive learning to train user embeddings to retrieve similar users and introduce collaborative information. For Challenge 2, we design a personalized retriever and reranker to retrieve the top-k documents from these users' histories. We take into account the user's preference during retrieval and reranking. Then we leverage feedback from the LLM to fine-tune the personalized retriever and reranker, enabling them to retrieve documents that meet the personalized generation needs of the LLM. Experimental results on the Language Model Personalization (LaMP) benchmark validate the effectiveness of CFRAG. Further analysis confirms the importance of incorporating collaborative information.

CCS Concepts

• **Information systems** → **Personalization**; • **Computing methodologies** → **Natural language generation**.

*Jun Xu is the corresponding author. Work partially done at Engineering Research Center of Next-Generation Intelligent Search and Recommendation, Ministry of Education. Work done when Teng Shi was an intern at Kuaishou.

SIGIR '25, Padua, Italy

ACM ISBN 979-8-4007-1592-1/2025/07
https://doi.org/10.1145/3726302.3730075

Keywords

Large language models; Retrieval augmented generation

ACM Reference Format:
Teng Shi, Jun Xu, Xiao Zhang, Xiaoxue Zang, Kai Zheng, Yang Song, and Han Li. 2025. Retrieval Augmented Generation with Collaborative Filtering for Personalized Text Generation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3726302.3730075

1 Introduction

Personalizing Large Language Models (LLMs) [56] to generate personalized outputs tailored to individual user preferences has emerged as a significant and rapidly growing field [16, 23, 29, 31, 32, 37, 38, 58]. Personalized Retrieval-Augmented Generation (RAG) [8] has become a commonly used approach for personalizing LLMs [29, 31, 32, 58].

The process of existing personalized RAG methods typically involves retrieving similar documents from the user's historical behaviors based on the user's input query, then concatenating these documents with the query as a prompt input to the LLM for generation. Although effective, this approach is limited to retrieving only the current user's history, neglecting collaborative information. Users with similar histories tend to be more alike, and the information from these similar users can also aid in personalizing generation for the current user. As shown in the example in Figure 1, the upper part illustrates the results of the existing RAG method, which retrieves documents from the current user's history. We can only infer from these results that "She" in the user's input refers to "Hillary Clinton". In contrast, the lower part demonstrates our method, which retrieves documents from the history of similar users. In this case, we can further infer that "his" in the user's input refers to "Donald Trump", leading to a better generation result. From this example, we can see that incorporating collaborative information allows the retrieval of more diverse documents, helping the LLM generate results that better meet the user's needs.

Inspired by the application of collaborative filtering in recommender systems [11, 41, 47], we propose to adapt collaborative information into RAG to personalize LLMs. However, adapting collaborative filtering to personalized RAG presents two challenges. **Challenge 1**: How to incorporate collaborative information. Without explicit labels indicating which users are similar, which users' information should be selected to help personalize generation for the current user? **Challenge 2**: How to retrieve documents that

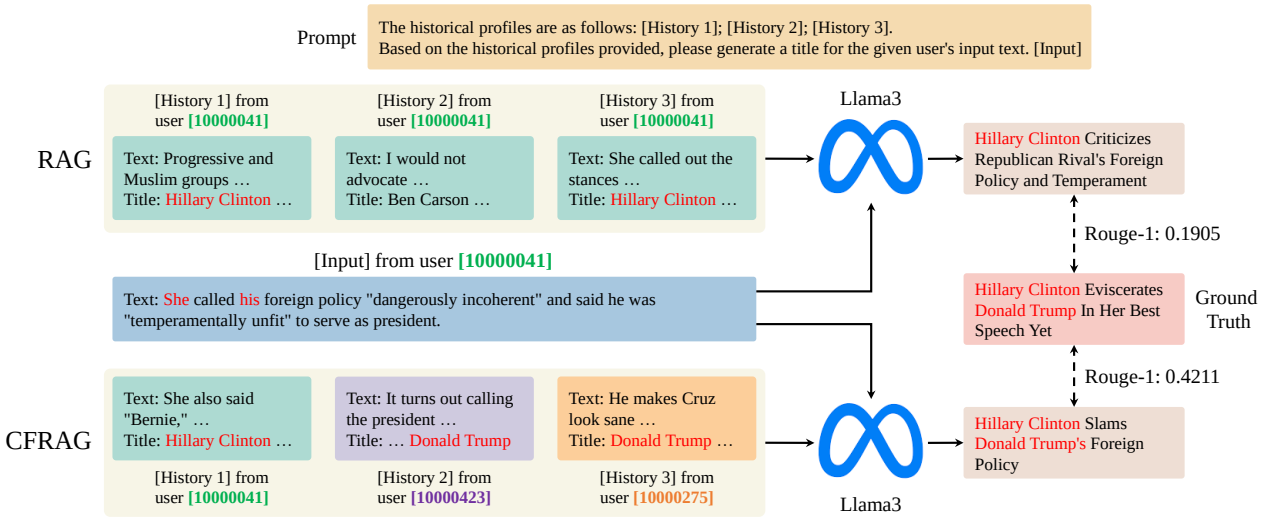


Figure 1: An example from the LaMP-4 dataset [32]. The task of LaMP-4 is to generate personalized news headlines based on user input. This example illustrates the benefit of collaborative information for LLM personalization: (a) The top shows results retrieved by the existing RAG method from the current user's history, where we can only infer that "She" in the user's input refers to "Hillary Clinton"‘. (b) The bottom shows results retrieved by our method from similar users' histories, allowing us to infer further that "his" in the user's input refers to "Donald Trump" thus enabling the generation of a more accurate result.

support personalized LLM generation, rather than relying on traditional semantic relevance? Pre-trained dense retrieval models [55] only retrieve based on the semantic relevance between the query and document. Directly using these models for retrieval may not necessarily result in content that allows the LLM to generate outputs that meet the user's needs [25, 36].

To address the above challenges, this paper proposes a method named **CFRAG** which adapts **C**ollaborative **F**iltering to personalized **R**etrieval **A**ugmented **G**eneration. Firstly, to address Challenge 1, since there are no explicit user similarity labels, we use contrastive learning [15, 45] to train user embeddings for retrieving similar users to introduce collaborative information. Specifically, we apply different data augmentation methods to the user's history to obtain different views, and then treat different views of the same user's history as positive samples for each other. Then we use contrastive learning on different views to train the user embeddings. Secondly, for Challenge 2, we designed a personalized retriever and reranker to retrieve the top-k documents from the histories of the retrieved users. In both retrieval and reranking, in addition to the semantic relevance between the query and documents, we also considered the user's preferences for different documents to enable personalized retrieval. Additionally, we further fine-tune the retriever and reranker based on the feedback from the LLM to ensure that the retrieved documents better support the personalized LLM generation. Finally, the top-k documents are concatenated with the user's input query to form a prompt, which is then fed into the LLM for personalized generation.

The major contributions of the paper are summarized as follows:
• We analyzed the necessity of introducing collaborative filtering into RAG for LLM personalization and identified the challenges: how to introduce collaborative information and how to retrieve documents that support personalized LLM generation.
• We proposed a method called CFRAG, which uses contrastive learning to train user embeddings for retrieving similar users and incorporating collaborative information. It leverages LLM feedback to train the personalized retriever and reranker, enabling them to retrieve documents that support personalized LLM generation.
• Experimental results on the Language Model Personalization (LaMP) [32] benchmark validate the effectiveness of CFRAG. The experimental analysis also demonstrates the importance of leveraging collaborative information.

2 Related Work

Personalization of LLMs. Large Language Models (LLMs) [56] have demonstrated remarkable capabilities in various fields, such as text generation [22], information retrieval [57], recommender systems [5, 42], and so on. However, since LLMs are typically designed to serve all tasks with a single model and are trained on broad, domain-agnostic data, they face challenges in adapting to the personalized needs of individual users [4, 32]. Therefore, LLM personalization has attracted widespread attention [16, 31, 58].

Existing works on LLM personalization mainly include the following types of methods: (1) Fine-tuning a personalized LLM for each user [37, 38, 43]; Tan et al. [38] fine-tuned the LLM using LoRA [12] to get personalized LoRA parameters for each user. (2) Aligning LLMs with user-specific preferences through Reinforcement Learning from Human Feedback (RLHF) [16, 23, 44]; Jang et al. [16] first trained different parameters for various objectives using RLHF, then merged these parameters based on users' personalized needs. (3) Incorporating user-specific context into the prompt [21, 27, 29, 31, 32, 58]. Richardson et al. [29] used instruction-tuned LLMs to summarize user history and then incorporated it into prompts for generation. Salemi et al. [31, 32] used RAG to retrieve relevant documents from user history based on the input query and incorporated them into the prompt.

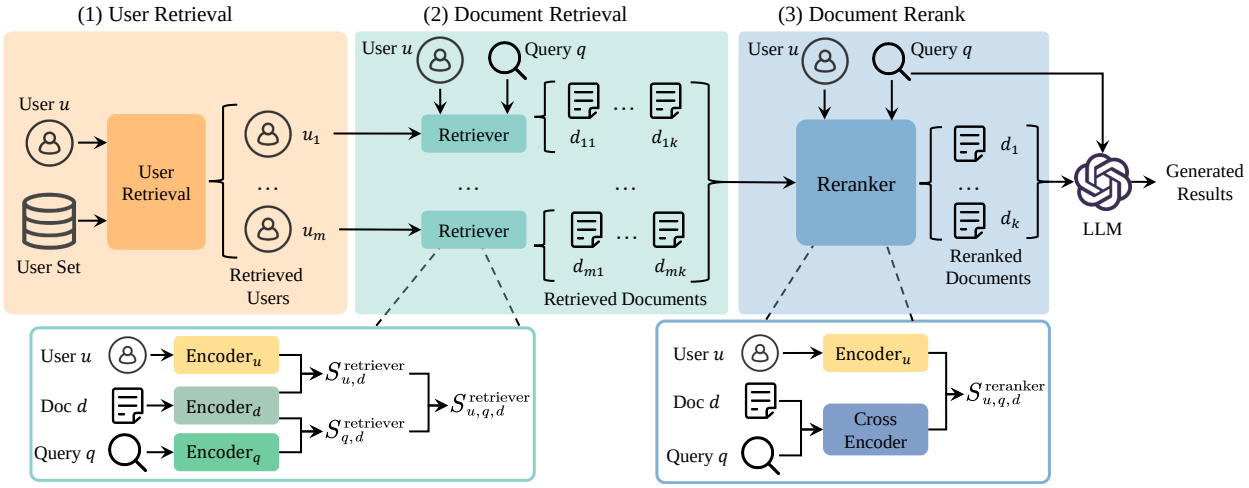


Figure 2: The architecture of CFRAG. From left to right: (a) User Retrieval retrieves similar users (Section 4.1); (b) Retriever retrieves the top-k documents from each user's history (Section 4.2); (c) Reranker reranks the $m \times k$ documents to get the final top-k documents, which are then concatenated with the query and input into the LLM for personalized text generation (Section 4.3).

This paper further introduces collaborative filtering for personalization based on the RAG framework. Collaborative filtering has already been applied in fields such as recommender systems [33–35, 39, 49–53] and has been proven effective. It assumes that users who have interacted with similar items share similar preferences, and recommending items from similar users to the current user can meet their needs. Some works [11, 47] learn the collaborative information between users and items through matrix factorization [19], while others [10, 41] further explore higher-order collaborative information between users and items using graph neural networks. The application of collaborative filtering in LLM personalization remains under-explored.

Retrieval Augmented Generation. Retrieval Augmented Generation [7, 8] introduces external knowledge through document retrieval, alleviating issues such as LLM hallucinations [54], and enhancing LLMs' capabilities in knowledge-intensive tasks [17] such as open-domain question answering [14, 20]. Some works [3, 13] encode retrieved documents using separate encoders, and then fuse the results with the language model using cross-attention. A more common approach is to directly include the retrieved documents in the prompt of the LLM [2, 9, 20, 25, 36]. In recent years, this in-context RAG framework has also been applied to LLM personalization, which is personalized by retrieving documents from the user's history [31, 32, 58]. This paper introduces collaborative filtering by retrieving similar users' histories for better personalization.

3 Problem Formulation

Let $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$ denotes the set of all users, where M is the number of users. Each user $u \in \mathcal{U}$ has a chronologically ordered history $\mathcal{H}_u = [d_1, d_2, \ldots, d_N]$ which includes all her historical documents, where N is the number of documents in the history. The personalized text generation dataset is $\mathcal{D} = \{(u, q, y)_i\}_{i=1}^{|\mathcal{D}|}$. For each instance, q is the query input by the user u to the LLM, and y is the target output. Our goal is first to introduce collaborative information by retrieving the top-m most similar users for user u:

$$\mathcal{U}_{\text{retrieved}} = \{u_1, u_2, \ldots, u_m\}.$$

Then, we use a retriever to retrieve the top-k documents from each of the m users' histories, resulting in a total of $m \times k$ documents.

$$\mathcal{D}_{\text{retrieved}} = \{d_{i,j} | i \in \{1, \ldots, m\}, j \in \{1, \ldots, k\}\}.$$

Finally, we use a reranker to rerank these $m \times k$ documents and obtain the final top-k documents:

$$\mathcal{D}_{\text{reranked}} = \{d_i | i \in \{1, \ldots, k\}\}.$$

These top-k documents will be concatenated with the user's query q as a prompt and input into the LLM, enabling it to generate a response that aligns with the target output y.

This paper primarily focuses on how to retrieve $\mathcal{U}_{\text{retrieved}}$ to introduce collaborative information, and how to train the retriever and reranker so that they can effectively retrieve documents that support the personalized LLM generation.

4 Our Approach

This section introduces our method CFRAG. CFRAG's overall architecture is shown in Figure 2. As mentioned in Section 1, to address Challenge 1, i.e., how to introduce collaborative information, we first train user embeddings using contrastive learning to retrieve the top-m most similar users (see Section 4.1). For Challenge 2, which involves retrieving documents that support personalized LLM generation, we fine-tune the personalized retriever and reranker using LLM feedback. The retriever first retrieves the top-k documents from the history of each of the m users, resulting in $m \times k$ documents (see Section 4.2). The reranker then reranks these documents to obtain the final top-k documents as input for the LLM (see Section 4.3).

4.1 User Retrieval

First, we perform user retrieval to get the top-m most similar users for user u to introduce collaborative information. However, we do

not have labels indicating which users are similar to each other. To address this, we employ a contrastive learning [15, 45] approach. We apply different data augmentation methods to the user history $\mathcal{H}_u$ to obtain different views of the user's history. We treat different views of the same user as positive samples and the histories of other users as negative samples, and then we use the InfoNCE [28] loss to train user embeddings for retrieval. Figure 3 illustrates the process of training user embeddings using contrastive learning.

4.1.1 User Encoder. Specifically, we first use an embedding model (such as BERT [6], RoBERTa [26], BGE [46] etc.) $\mathbf{Emb}(\cdot)$ to encode each document in the user's history $\mathcal{H}_u$ to obtain $\mathbf{E}_u = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N]^\top \in \mathbb{R}^{N\times d}$, where $\mathbf{e}_i = \mathbf{Emb}(d_i)$ and d is the embedding dimension. To model the sequential relationships between different documents in the user's history, we introduce positional embedding $\mathbf{P} \in \mathbb{R}^{N\times d}$. Afterward, the history $\mathcal{H}_u$'s embedding becomes $\widehat{\mathbf{E}}_u = \mathbf{E}_u + \mathbf{P}$. Then, we apply a transformer [40] as the user encoder to encode the user's history $\widehat{\mathbf{E}}_u$ and average the transformer's output to obtain the user's embedding:

$$\mathbf{e}_u = \text{Encoder}_u(u) = \text{MEAN}(\text{Trm}(\widehat{\mathbf{E}}_u)) \in \mathbb{R}^d, \tag{1}$$

where $\text{Encoder}_u(\cdot) \to \mathbb{R}^d$ denotes the user encoder, $\text{Trm}(\cdot)$ denotes a transformer encoder. Next, we train the transformer encoder using contrastive learning.

4.1.2 Data Augmentation. We generate different views of $\mathcal{H}_u$ using the following three data augmentation methods:

Document Crop. We randomly select a continuous sub-sequence of length $L_c = \lfloor \eta_c N \rfloor$ from $\mathcal{H}_u$, where η_c is a hyper-parameter controlling the crop ratio. The history after cropping is as follows:

$$\mathcal{H}_u^{\text{crop}} = [d_c, d_{c+1}, \ldots, d_{c+L_c-1}].$$

Document Mask. For the history $\mathcal{H}_u$, we randomly mask out $L_m = \lfloor \eta_m N \rfloor$ documents $\mathcal{I}_{\text{mask}} = \{i_1, i_2, \ldots, i_{L_m}\}$, where $\mathcal{I}_{\text{mask}}$ is the set of indices corresponding to the masked documents and η_m is a hyper-parameter that controls the mask ratio. The masked documents are replaced with a special token [mask]. The history after masking is as follows:

$$\mathcal{H}_u^{\text{mask}} = [\hat{d}_1, \hat{d}_2, \ldots, \hat{d}_N],$$
$$\hat{d}_i = \begin{cases} d_i, & i \notin \mathcal{I}_{\text{mask}}, \\ [\text{mask}], & i \in \mathcal{I}_{\text{mask}}. \end{cases}$$

Document Reorder. We randomly select a sub-sequence $[d_r, d_{r+1}, \ldots, d_{r+L_r-1}]$ of length $L_r = \lfloor \eta_r N \rfloor$ from $\mathcal{H}_u$, where η_r is a hyper-parameter controlling the reorder ratio, and then randomly shuffle the order of the documents within the sub-sequence to obtain $[\hat{d}_r, \hat{d}_{r+1}, \ldots, \hat{d}_{r+L_r-1}]$. The history after reordering is as follows:

$$\mathcal{H}_u^{\text{reorder}} = [d_1, d_2, \ldots, \hat{d}_r, \ldots, \hat{d}_{r+L_r-1}, \ldots, d_N].$$

4.1.3 Contrastive Loss. Each time, we randomly select two data augmentation methods $\mathcal{A}'$ and $\mathcal{A}''$ to generate two different views of $\mathcal{H}_u$, denoted as $\mathcal{H}'_u$ and $\mathcal{H}''_u$. Then, using the encoder described in Section 4.1.1, we obtain the user embeddings $\mathbf{e}'_u$ and $\mathbf{e}''_u$ corresponding to the different views. Since $\mathbf{e}'_u$ and $\mathbf{e}''_u$ are obtained through data augmentation of $\mathcal{H}_u$, they are more similar to each other. Therefore, we treat them as positive samples for each other

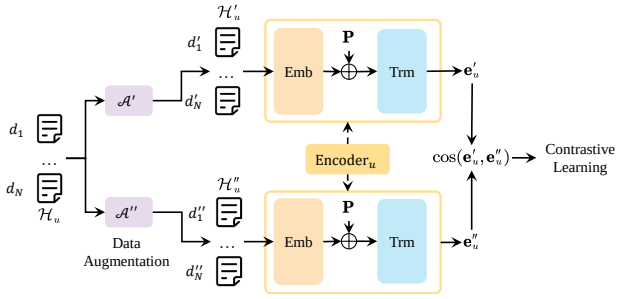


Figure 3: Contrastive learning for user embedding training.

and use the views generated from the augmented histories of other users in the same batch as negative samples. We then perform contrastive learning using the InfoNCE [28] loss as follows:

$$\mathcal{L}_{\text{CL}} = -\left[\log \frac{\exp(\cos(\mathbf{e}'_u, \mathbf{e}''_u)/\tau_1)}{\sum_{u^- \in \mathcal{U}_{\text{neg}}} \exp(\cos(\mathbf{e}'_u, \mathbf{e}''_{u^-})/\tau_1)} + \log \frac{\exp(\cos(\mathbf{e}'_u, \mathbf{e}''_u)/\tau_1)}{\sum_{u^- \in \mathcal{U}_{\text{neg}}} \exp(\cos(\mathbf{e}'_{u^-}, \mathbf{e}''_u)/\tau_1)}\right], \tag{2}$$

where τ_1 is the temperature coefficient, $\mathcal{U}_{\text{neg}}$ are the set of randomly sampled in-batch negative samples, and $\cos(\cdot)$ denotes the cosine similarity.

4.1.4 Top-m User Retrieval. After training with contrastive learning, we can use the encoder from Section 4.1.1 to obtain the user embedding $\mathbf{e}_u$. We then calculate the cosine similarity between each pair of user embeddings and retrieve the top-m most similar users $\mathcal{U}_{\text{retrieved}} = \{u_1, u_2, \ldots, u_m\}$ for user u. Subsequently, the histories of these m users will be used for further document retrieval.

4.2 Document Retrieval

After retrieving the top-m users, we design a personalized retriever to retrieve the top-k documents from each user's history, resulting in a total of $m \times k$ candidate documents $\mathcal{D}_{\text{retrieved}} = \{d_{i,j} | i \in \{1, \ldots, m\}, j \in \{1, \ldots, k\}\}$. This section introduces how the retriever is designed and how it's trained to retrieve documents that better align with the requirements of personalized LLM generation.

4.2.1 Retriever. First, we use a pre-trained dense retrieval model (such as BGE retriever [46]) to compute the semantic relevance between the query and the candidate documents:

$$S_{q,d}^{\text{retriever}} = \cos(\text{Encoder}_q(q), \text{Encoder}_d(d)), \tag{3}$$

where $\text{Encoder}_q(\cdot) \to \mathbb{R}^d$ and $\text{Encoder}_d(\cdot) \to \mathbb{R}^d$ are the encoders for the query and the document in the retrieval model, respectively. Pre-trained retrieval models typically use $S_{q,d}^{\text{retriever}}$ directly for retrieval. However, $S_{q,d}^{\text{retriever}}$ only considers the semantic relevance between the query and the document. Since different users might input the same query but expect different outputs due to their varying preferences, we further account for user personalization by calculating the preference score of the user for the document as follows:

$$S_{u,d}^{\text{retriever}} = \cos(\text{MLP}_1(\mathbf{e}_u), \text{Encoder}_d(d)), \tag{4}$$

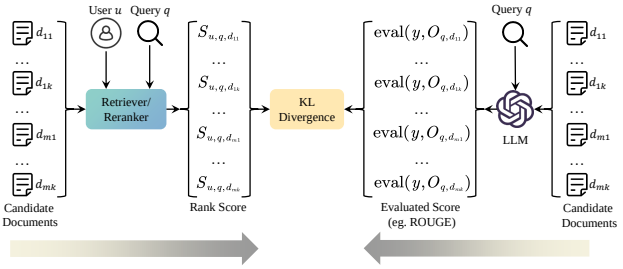


Figure 4: The method of training the retriever and reranker using LLM feedback.

where $\text{MLP}_1 : \mathbb{R}^d \to \mathbb{R}^d$ is a multi-layer perceptron that maps the user embedding to the space where the cosine similarity is computed. $\mathbf{e}_u$ is the embedding obtained in Section 4.1.1. The total score for retrieval is computed as follows:

$$S^{\text{retriever}}_{u,q,d} = (1-\alpha)S^{\text{retriever}}_{q,d} + \alpha S^{\text{retriever}}_{u,d}, \tag{5}$$

where α is a hyper-parameter that controls the weight of personalization.

4.2.2 Training. Since the pre-trained dense retrieval model is not fine-tuned for our specific task, the retrieved results may not necessarily lead to LLM responses that better match the target output y [25, 36]. However, there is no ground truth indicating which documents are better. Therefore, we evaluate the difference between the LLM's output and the target output y, using this as a label to train the retrieval model. Figure 4 shows the process of training the retriever using LLM feedback.

Specifically, we first use the pre-trained retrieval model to retrieve the top-k documents from each of the m users' histories based on $S^{\text{retriever}}_{q,d}$ in Eq. (3), resulting in a total of $m \times k$ candidate documents. These documents are then concatenated with the query one by one and used as prompts for the LLM, producing $m \times k$ outputs:

$$\{O_{q,d_{i,j}} = \text{LLM}(q, d_{i,j}) | i \in \{1, \ldots, m\}, j \in \{1, \ldots, k\}\},$$

where $\text{LLM}(q, d_{i,j})$ represents the output generated by inputting the concatenated query q and document $d_{i,j}$ into the LLM. Then, based on the quality of these outputs, we can calculate the distribution of these candidate documents as follows:

$$p_{\text{LLM}}(d_{i,j}|q,y) = \frac{\exp(\text{eval}(y, O_{q,d_{i,j}}))}{\sum_{i=1}^{m}\sum_{j=1}^{k}\exp(\text{eval}(y, O_{q,d_{i,j}}))}, \tag{6}$$

where $\text{eval}(\cdot)$ measures the difference between the target output y and the LLM's output, using metrics such as ROUGE [24] score. A larger value returned by $\text{eval}(\cdot)$ indicates a better-generated result. Similarly, we can also calculate the score distribution of the candidate documents by the retrieval model based on $S^{\text{retriever}}_{u,q,d}$ in Eq. (5):

$$p_{\text{retriever}}(d_{i,j}|q,u) = \frac{\exp(S^{\text{retriever}}_{u,q,d_{i,j}})}{\sum_{i=1}^{m}\sum_{j=1}^{k}\exp(S^{\text{retriever}}_{u,q,d_{i,j}})}. \tag{7}$$

We aim for the retrieval model to retrieve documents that lead to better LLM-generated results, which means making the distribution $p_{\text{retriever}}(d|q,u)$ in Eq. (7) closer to the distribution $p_{\text{LLM}}(d|q,y)$ in Eq (6). Therefore, we compute the KL divergence between the two distributions as the loss to optimize the retriever:

$$\mathcal{L}_{\text{retriever}} = \text{KL}(p_{\text{retriever}}(d|q,u) \,||\, p_{\text{LLM}}(d|q,y)). \tag{8}$$

4.3 Document Rerank

After retrieving $\mathcal{D}_{\text{retrieved}}$ through the retriever, in this section, we further refine the results by reranking $\mathcal{D}_{\text{retrieved}}$ to obtain the final top-k ranked results $\mathcal{D}_{\text{reranked}} = \{d_i | i \in \{1, \ldots, k\}\}$.

4.3.1 Reranker. We use a pre-trained cross-encoder (such as the BGE reranker [46]) to encode the query and document, obtaining the hidden state corresponding to the [CLS] token from the last layer:

$$\mathbf{h}_{q,d} = \text{CrossEncoder}(q, d), \tag{9}$$

where $\mathbf{h}_{q,d} \in \mathbb{R}^d$. Similarly, when reranking, in addition to considering the semantic relevance between query and document, we also take into account the user's personalized preferences. However, since the cross-encoder does not encode documents separately, it cannot compute the cosine similarity between users and documents as shown in Eq. (4) to express the user preference score. Therefore, we directly concatenate the user embeddings to the output of the cross-encoder to account for the influence of user preferences. The overall score used for reranking is calculated as follows:

$$S^{\text{reranker}}_{u,q,d} = \text{MLP}_3(\text{CONCAT}(\mathbf{h}_{q,d}, \text{MLP}_2(\mathbf{e}_u))), \tag{10}$$

where $\text{MLP}_2 : \mathbb{R}^d \to \mathbb{R}^d$ and $\text{MLP}_3 : \mathbb{R}^{2d} \to \mathbb{R}$ are two multi-layer perceptions. $\text{CONCAT}(\cdot)$ denotes the concatenation operation.

4.3.2 Training. Similar to the retriever's training in Section 4.2.2, we also want the reranker to assign higher scores to the documents that lead to better LLM-generated results. Therefore, we train the reranker using a similar approach.

We use the trained retrieval model from Section 4.2.2 to retrieve top-k documents from the history of each of the m users, resulting in a total of $m \times k$ candidate documents. These documents are concatenated with the query q and used as prompts for the LLM, producing $m \times k$ outputs. Similar to Eq.(6), we can obtain the distribution $p_{\text{LLM}}(d|q,y)$ of these candidate documents. Based on $S^{\text{reranker}}_{u,q,d}$ in Eq. (10), we can also get the score distribution of the candidate documents by the reranker:

$$p_{\text{reranker}}(d_{i,j}|q,u) = \frac{\exp(S^{\text{reranker}}_{u,q,d_{i,j}})}{\sum_{i=1}^{m}\sum_{j=1}^{k}\exp(S^{\text{reranker}}_{u,q,d_{i,j}})}. \tag{11}$$

We compute the KL divergence between distributions $p_{\text{reranker}}(d|q,u)$ and $p_{\text{LLM}}(d|q,y)$ as the loss to optimize the reranker:

$$\mathcal{L}_{\text{reranker}} = \text{KL}(p_{\text{reranker}}(d|q,u) \,||\, p_{\text{LLM}}(d|q,y)). \tag{12}$$

The loss allows the reranker to assign higher scores to documents that enable better personalized generation by the LLM.

4.4 Discussion

Computational Efficiency. CFRAG comprises three modules. The User Encoder is a lightweight, single-layer Transformer with inputs derived from a frozen BGE embedding (dimension 768), resulting in minimal parameter overhead. The retriever and reranker are comparable in size to BERT (approximately 100M parameters). Overall,

Table 1: Statistics of the datasets used in this paper.

Dataset	LaMP-1	LaMP-2	LaMP-3	LaMP-4	LaMP-5	LaMP-7
#Users	6,542	929	20,000	1,643	14,682	13,437
#Train	6,542	5,073	20,000	12,500	14,682	13,437
#Dev	1,500	1,410	2,500	1,500	1,500	1,498
#Test	1,500	1,557	2,500	1,800	1,500	1,500

the training cost is low due to the modest parameter size. During inference, user and document embeddings can be precomputed, requiring only similarity calculations for retrieval, ensuring minimal computational cost. This efficiency enables our method to generalize quickly to new datasets.

5 Experiments

We conducted experiments to evaluate the performance of CFRAG. The source code is available. [1]

5.1 Experimental Setup

5.1.1 Dataset. We conducted experiments on the Language Model Personalization (LaMP) [32] benchmark, which consists of seven personalized text generation tasks. We excluded LaMP-6 because its data is not publicly available. The remaining tasks include: **LaMP-1** (Personalized Citation Identification); **LaMP-2** (Personalized Movie Tagging); **LaMP-3** (Personalized Product Rating); **LaMP-4** (Personalized News Headline Generation); **LaMP-5** (Personalized Scholarly Title Generation); **LaMP-7** (Personalized Tweet Paraphrasing). We used the time-based split provided by LaMP to divide the data into training, validation, and test sets. The statistics of these datasets are shown in Table 1.

5.1.2 Evaluation Metrics. Following previous works [31, 32], we evaluate Accuracy and F-1 score for LaMP-1 and LaMP-2, mean absolute error (MAE) and root mean squared error (RMSE) for LaMP-3, ROUGE-1 and ROUGE-L [24] for LaMP-4, LaMP-5 and LaMP-7.

5.1.3 Baselines. In this work, we compare CFRAG with the following methods.

No Personalization: We directly input the user's query into the LLM without retrieving from user history, using this as the non-personalized baseline. We refer to this method as **Zero Shot**.

Personalized Baselines: We compared CFRAG with methods that personalize by retrieving from user history using different retrieval models, including: (1) **Random** selects k items randomly from the user's history; (2) **Recency** selects the most recent k items from the user's history; (3) **BM25** [30] retrieves top-k items from the user's history using BM25; (4) **BGE** [46] retrieves top-k items from the user's history using BGE retriever; (5) **ROPG** [31] optimizes the dense retrieval model based on the results generated by the LLM.

5.1.4 Implementation Details. We conducted experiments on two LLMs: Llama3-8B-Instruct [1] and Qwen2-7B-Instruct [48]. In this paper, we do not fine-tune the LLM because fine-tuning is costly and could cause the LLM to retain user information, potentially compromising user privacy. To ensure a fair comparison, we use greedy search for text generation. The dense retrieval model used in all methods is bge-base-en-v1.5[2] [46]. The cross-encoder used for reranker in Section 4.3.1 is bge-reranker-base[3] [46]. All hyperparameters for the baselines are searched according to the settings in the original papers. The embedding dimension d is set to 768. The number of retrieved documents k is set to 5, and the number of retrieved users m is tuned among $\{2, 3, 4, 5, 6\}$. The $\mathrm{Trm}(\cdot)$ encoder in Eq. (1) has 1 layer and 2 heads. The hyperparameters L_c, L_m, and L_r used for data augmentation in Section 4.1.2 are set to 0.7, 0.3, and 0.3, respectively. The temperature parameters τ_1 in Eq. (2) is tuned among $\{0.01, 0.1, 1\}$. The weight α in Eq. (5) is tuned among $[0.01, 1.0]$. The learning rate is tuned among $\{1e\text{-}3, 1e\text{-}4, 1e\text{-}5\}$. Adam [18] is used to conduct the optimization. The data input and output formats are provided in Appendix A.

5.2 Experimental Results

Experimental results are shown in Table 2. From the results, we can find that:

• Firstly, compared to existing methods, CFRAG achieved the best results across six datasets in the LaMP benchmark. This demonstrates the effectiveness of introducing collaborative information between users into RAG and using LLM feedback to tune the retriever and reranker to ensure that they can retrieve the documents that support the personalized LLM generation.

• Secondly, we can observe that even randomly selecting user history outperforms the zero-shot method without any user history. This highlights the importance of incorporating user history to reflect user preferences for personalized generation. Additionally, we observe that retrieval methods perform better than simply selecting the most recent user history, underscoring the importance of retrieval.

• Thirdly, we also observe that, in most cases, RAG and ROPG methods using dense retrieval models outperform BM25. Additionally, CFRAG, which fine-tunes the retriever based on LLM feedback, achieves better results. This shows, on the one hand, that the better the retriever, the better the generation results, and on the other hand, fine-tuning the retriever based on LLM feedback to ensure it can retrieve the documents that meet the personalized generation needs of LLM is crucial.

5.3 Ablation Study

We conducted an ablation study to investigate the effectiveness of different modules in CFRAG, as shown in Table 3. CFRAG consists of three modules: User Retrieval, Document Retrieval, and Document Rerank. We removed different modules from CFRAG one by one to verify the effectiveness of each module.

5.3.1 User Retrieval. First, we validated the effectiveness of introducing collaborative information by retrieving similar users, as shown in row (1) of Table 3. It can be seen that without retrieving similar users and only retrieving from the current user's history, the performance is worse than that of CFRAG, highlighting the importance of collaborative information.

We also validated the effectiveness of training user embeddings using contrastive learning. For comparison, we directly averaged

[1] https://github.com/TengShi-RUC/CFRAG

[2] https://huggingface.co/BAAI/bge-base-en-v1.5

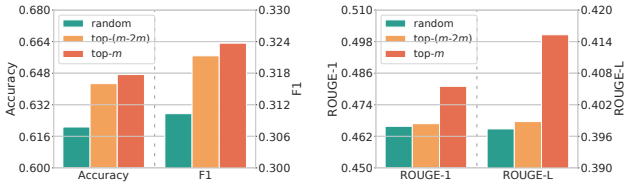
[3] https://huggingface.co/BAAI/bge-reranker-base

Table 2: Comparison of the performance of CFRAG with other approaches on the LaMP benchmark. ↑ indicates that a higher value for the corresponding metric is better, while ↓ indicates that a lower value is better. The best and the second-best methods are highlighted in bold and underlined fonts, respectively. "*" indicates improvements over the second-best methods are statistically significant (t-test, p-value< 0.05).

LLMs	Retrievers	LaMP-1		LaMP-2		LaMP-3		LaMP-4		LaMP-5		LaMP-7	
		Accuracy ↑	F1 ↑	Accuracy ↑	F1 ↑	MAE ↓	RMSE ↓	ROUGE-1 ↑	ROUGE-L ↑	ROUGE-1 ↑	ROUGE-L ↑	ROUGE-1 ↑	ROUGE-L ↑
Llama3	Zero Shot	0.4993	0.2497	0.2993	0.0200	0.5024	0.7904	0.1406	0.1228	0.4417	0.3650	0.3079	0.2593
	Random	0.5740	0.2870	0.3929	0.0262	0.4104	0.7833	0.1787	0.1571	0.4533	0.3875	0.3137	0.2508
	Recency	0.6040	0.3020	0.3993	0.0266	0.3980	0.7491	0.1856	0.1650	0.4573	0.3928	0.3325	0.2686
	BM25 [30]	0.6240	0.3120	0.4255	0.0284	0.4060	0.7666	0.1803	0.1591	0.4637	0.3978	0.3449	0.2780
	BGE [46]	0.6327	0.3163	0.4574	0.0305	0.3528	0.6969	0.1811	0.1611	0.4638	0.3958	0.3391	0.2742
	ROPG [31]	0.6440	0.3220	0.4681	0.0312	0.3456	0.6922	0.1838	0.1634	0.4638	0.3956	0.3530	0.2881
	CFRAG	**0.6533***	**0.3267***	**0.5340***	**0.0356***	**0.2812***	**0.5997***	**0.1957***	**0.1745***	**0.4810***	**0.4153***	**0.3752***	**0.3055***
Qwen2	Zero Shot	0.5000	0.2500	0.2908	0.0194	0.4444	0.7805	0.1264	0.1081	0.4144	0.3468	0.3972	0.3229
	Random	0.5633	0.2817	0.3284	0.0219	0.4000	0.7621	0.1581	0.1377	0.4580	0.3921	0.4291	0.3564
	Recency	0.5773	0.2887	0.3326	0.0222	0.3912	0.7563	0.1581	0.1369	0.4562	0.3913	0.4247	0.3525
	BM25 [30]	0.5987	0.2993	0.3532	0.0235	0.4228	0.8027	0.1580	0.1374	0.4613	0.3950	0.4290	0.3570
	BGE [46]	0.6080	0.3040	0.3674	0.0245	0.3696	0.7211	0.1613	0.1398	0.4571	0.3910	0.4347	0.3605
	ROPG [31]	0.6093	0.3047	0.3830	0.0255	0.3672	0.7332	0.1617	0.1401	0.4600	0.3946	0.4345	0.3610
	CFRAG	**0.6133**	**0.3067**	**0.3957***	**0.0264**	**0.3536***	**0.7071***	**0.1621**	**0.1412**	**0.4703***	**0.4029***	**0.4425***	**0.3708***

Table 3: Ablation Study of CFRAG on LaMP based on Llama3. "MEAN" represents using the average of user history document embeddings as the user embedding. "w/o" indicates the corresponding module in CFRAG is removed.

Variants		LaMP-1		LaMP-2		LaMP-3		LaMP-4		LaMP-5		LaMP-7	
#	Model	Accuracy ↑	F1 ↑	Accuracy ↑	F1 ↑	MAE ↓	RMSE ↓	ROUGE-1 ↑	ROUGE-L ↑	ROUGE-1 ↑	ROUGE-L ↑	ROUGE-1 ↑	ROUGE-L ↑
(0)	**CFRAG**	**0.6533**	**0.3267**	**0.5340**	**0.0356**	**0.2812**	**0.5997**	**0.1957**	**0.1745**	**0.4810**	**0.4153**	**0.3752**	**0.3055**
(1)	w/o User Retrieval	0.6400	0.3200	0.4936	0.0329	0.3444	0.6925	0.1914	0.1689	0.4642	0.3963	0.3566	0.2903
(2)	User Retrieval (MEAN)	0.6420	0.3210	0.5064	0.0338	0.3412	0.6867	0.1847	0.1639	0.4779	0.4113	0.3722	0.3022
(3)	w/o Retriever Tuning	0.6453	0.3227	0.4979	0.0332	0.2852	0.6070	0.1916	0.1704	0.4742	0.4048	0.3599	0.2940
(4)	w/o $S^{\text{retriever}}_{u,d}$ in Eq. (5)	0.6333	0.3167	0.5113	0.0341	0.3324	0.6861	0.1895	0.1696	0.4750	0.4088	0.3732	0.3039
(5)	w/o Reranker Tuning	0.6307	0.3153	0.4695	0.0313	0.3696	0.7392	0.1766	0.1550	0.4714	0.4068	0.3432	0.2775
(6)	w/o $\mathbf{e}_u$ in Eq. (10)	0.6313	0.3157	0.4993	0.0333	0.3420	0.6925	0.1887	0.1672	0.4772	0.4123	0.3731	0.3030



(a) LaMP-1 (b) LaMP-5

Figure 5: Results of using different methods to select users for introducing collaborative information. "random" indicates randomly selecting m users; "top-(m-2m)" represents selecting users whose similarity to the current user ranks between m and 2m; "top-m" indicates selecting the most similar m users.

the document embeddings from the user's history to create user embeddings for retrieval, as shown in row (2) of Table 3. It can be seen that CFRAG, which uses user embeddings trained with contrastive learning, achieves better results. This is because contrastive learning constructs user similarity labels through data augmentation and uses the InfoNCE loss to help the embeddings learn which users are similar. In contrast, using mean pooling directly cannot capture user similarity.

5.3.2 Document Retrieval. We also validated the effectiveness of the personalized retriever we designed, as shown in Table 3, rows (3) and (4). First, in row (3), we can see that without fine-tuning based on LLM feedback, using a pre-trained dense retrieval model leads to worse performance. This indicates that retrieval cannot be based solely on semantic relevance, ensuring that the retrieved documents support personalized LLM generation is crucial. Additionally, we analyzed the impact of removing $S^{\text{retriever}}_{u,d}$ from Eq. (4) and only using $S^{\text{retriever}}_{q,d}$ from Eq. (3) for retrieval, as indicated in row (4). The results decreased, demonstrating that users' personalized preferences should also be considered during retrieval, rather than solely focusing on the semantic relevance between the query and documents.

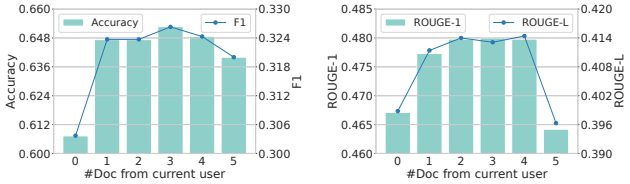
5.3.3 Document Rerank. We also validated the effectiveness of the personalized reranker we designed, as shown in Table 3, rows (5) and (6). First, in row (5), it can be seen that using a pre-trained reranker leads to worse results, highlighting the importance of fine-tuning based on LLM feedback. We also observed the effect of removing $\mathbf{e}_u$ from Eq. (10) and only using $\mathbf{h}_{q,d}$ to calculate $S^{\text{rerranker}}_{q,d}$ for ranking, as indicated in row (6). The results decreased in this case, highlighting the importance of considering users' personalized preferences in the reranker.

5.4 Experimental Analysis

As mentioned in Section 1, adapting collaborative filtering into personalized RAG faces two challenges. **Challenge 1**: How to introduce collaborative information? **Challenge 2**: How to retrieve

(a) LaMP-1 (b) LaMP-5

Figure 6: Results using different retrievers and rerankers. "BM25" indicates using BM25 as both the retriever and reranker, while "w/o Tuning" refers to using pre-trained retrievers and rerankers without LLM feedback fine-tuning.



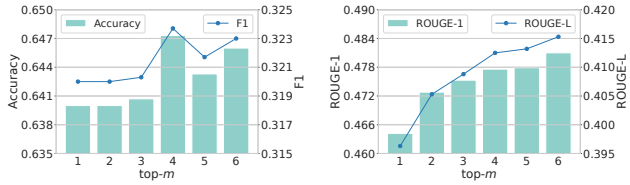
(a) LaMP-1 (b) LaMP-5

Figure 7: Performance under different numbers of retrieved documents from the current user u's history in the top-k documents.

documents that support personalized LLM generation? In this section, we conduct experimental analysis to further demonstrate the effectiveness of our method in addressing these two challenges. Additionally, we provide further analysis of the results of CFRAG and the impact of hyper-parameters. Due to space limitations, we conducted experimental analysis on the LaMP-1 and LaMP-5 datasets.

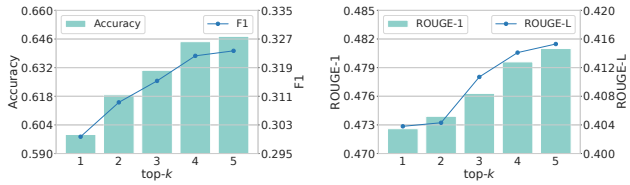
5.4.1 Effectiveness of User Retrieval using Contrastive Learning (Challenge 1). As described in Section 1, to address Challenge 1, we train user embeddings using contrastive learning to retrieve the top-m most similar users for introducing collaborative information. To validate the effectiveness of this approach, we compared it with randomly selecting m users and selecting users from top-m to $2m$, as shown in Figure 5. First, we can see that randomly selecting users yields the worst performance, indicating that collaborative information cannot be introduced indiscriminately. Secondly, the results show that retrieving users from the range of top-m to $2m$ performs worse than using the top-m users, suggesting that information from users who are more similar to the current user u is more important. These highlight the importance of retrieving the most similar top-m users

5.4.2 Effectiveness of Document Retrieval using LLM Feedback (Challenge 2). As mentioned in Section 1, to address Challenge 2, we fine-tune the retriever and reranker using feedback from the content generated by the LLM, enabling them to retrieve documents that better meet personalized LLM generation needs. To validate its effectiveness, we compared the results with those using retrievers and rerankers without LLM feedback fine-tuning, as well as using



(a) LaMP-1 (b) LaMP-5

Figure 8: Performance under different numbers of retrieved users. The performance is the worst since no collaborative information is introduced when $m = 1$.



(a) LaMP-1 (b) LaMP-5

Figure 9: Performance under different numbers of retrieved documents per user.

BM25 as the retriever and reranker, as shown in Figure 6. It can be observed that CFRAG performs the best, highlighting the importance of fine-tuning with LLM feedback rather than relying solely on semantic relevance.

5.4.3 Impact of the Number of Documents from the Current User. To further validate that CFRAG enhances personalization by incorporating collaborative information, we observed the impact of the number of documents from the current user in the final top-k documents on the results, as shown in Figure 7. We varied the number of documents retrieved from the current user's history in the top-k documents from 0 to 5, with the remaining documents retrieved from similar users' histories. The results indicate that retrieving only from the current user's history leads to poor performance, while appropriately retrieving documents from similar users' histories significantly improves the results. This verifies the importance of incorporating collaborative information.

5.4.4 Impact of the Number of Retrieved Users. Since we enhance personalized text generation by introducing collaborative filtering, we further explored how much collaborative information to introduce, specifically the impact of the number of retrieved users on the results, as shown in Figure 8. In LaMP-1, retrieving too few or too many users leads to poorer performance, with the best results at 4 users. In LaMP-5, the performance improves as the number of users increases. This highlights the importance of introducing collaborative filtering, but it also indicates that excessive introduction can lead to decreased effectiveness.

5.4.5 Impact of the Number of Retrieved Documents. We also analyzed the impact of the number of retrieved documents, k, on the results, as shown in Figure 9. It can be observed that as the number

Table 4: The format of input, output, and user history for different datasets in the LaMP [32] benchmark. In the input, *{history$_i$}* will be replaced by the retrieved *i*-th history, and each history is represented as shown in the "User History" column. The other *italicized text* in the input is replaced with the user's input. For text generation tasks, to ensure that the LLM does not generate irrelevant information, we instruct the LLM in the input to generate in JSON format, and then we extract the LLM's prediction from the JSON-formatted output.

Task	Input	Output	User History
LaMP-1	The historical profiles are as follows: *{history$_1$}* … *{history$_k$}*. Based on the historical profiles provided, please choose one of the following two references that is more relevant to the user's input title: [1] *{reference$_1$}*; [2] *{reference$_2$}*. Please just answer with "[1]" or "[2]" without explanation. "title": *{title}*.	[1]	"title": *{title}* "abstract": *{abstract}*
LaMP-2	The historical profiles are as follows: *{history$_1$}* … *{history$_k$}*. Based on the historical profiles provided, please select the tag from [sci-fi, based on a book, comedy …] that is most relevant to the user's input description. Please just answer with the tag name without explanation. "description": *{description}*; "tag":	comedy	"description": *{description}*; "tag": *{tag}*
LaMP-3	The historical profiles are as follows: *{history$_1$}* … *{history$_k$}*. Based on the historical profiles provided, what is the score of the following review on a scale of 1 to 5? just answer with 1, 2, 3, 4, or 5 without further explanation. "review": *{review}*; "score":	5	"review": *{review}* "score": *{score}*
LaMP-4	The historical profiles are as follows: *{history$_1$}* … *{history$_k$}*. Based on the historical profiles provided, please generate a title for the given user's input text. Please generate it in the following format: {"title": "generated title"} without explanation, and use only English. "text": *{text}*; "title":	{"title": Finding Happiness After Divorce – It Can Happen}	"text": *{text}* "title": *{title}*
LaMP-5	The historical profiles are as follows: *{history$_1$}* … *{history$_k$}*. Based on the historical profiles provided, please generate a title for the given user's input abstract. Please generate it in the following format: {"title": "generated title"} without explanation, and use only English. "abstract": *{abstract}*; "title":	{"title": Link-Reliability Based Two-Hop Routing for Wireless Sensor Networks.}	"abstract": *{abstract}* "title": *{title}*
LaMP-7	The historical profiles are as follows: *{history$_1$}* … *{history$_k$}*. Based on the style pattern of the historical tweets provided, please paraphrase the user's input tweet without any explanation before or after it. Please generate it in the following format: {"tweet": "generated tweet"} without explanation, and use only English. "tweet": *{tweet}*.	{"tweet":lilxcutiesworld the danny picture is GOOD!! I really like it.}	"tweet": *{tweet}*

of retrieved documents increases, performance improves, indicating the importance of retrieving user history to reflect user preferences for enhancing LLM-generated results. Since more documents lead to longer prompts and slower LLM generation, we chose $k = 5$ for our experiments.

6 Conclusion

In this paper, we propose CFRAG, which adapts collaborative filtering into RAG to personalize LLMs. To introduce collaborative information without explicit user labels and retrieve documents that support personalized LLM generation, we first train user embeddings through contrastive learning to retrieve similar users. Then, we design the personalized retriever and reranker that considers user preferences during retrieval and fine-tune them using LLM feedback. The results on the Language Model Personalization (LaMP) benchmark validate the effectiveness of CFRAG. The experimental analysis also confirms the effectiveness of each module within CFRAG.

A Appendix: Prompts

We provide detailed formats for the inputs, outputs, and user histories for the LLM across different datasets, as shown in Table 4.

Acknowledgments

This work was funded by the National Key R&D Program of China (2023YFA1008704), the National Natural Science Foundation of China (No. 62472426), Beijing Key Laboratory of Big Data Management and Analysis Methods, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, PCC@RUC, funds for building world-class universities (disciplines) of Renmin University of China, Kuaishou Technology.

References

[1] AI@Meta. 2024. Llama 3 Model Card. (2024). https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md

[2] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. [n. d.]. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. In *The Twelfth International Conference on Learning Representations*.

[3] Sebastian Borgeaud, Arthur Mensch, et al. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*. PMLR, 2206–2240.

[4] Jin Chen, Zheng Liu, et al. 2024. When large language models meet personalization: Perspectives of challenges and opportunities. *World Wide Web* 27, 4 (2024), 42.

[5] Sunhao Dai, Ninglu Shao, et al. 2023. Uncovering chatgpt's capabilities in recommender systems. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 1126–1132.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

[7] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A Survey on RAG Meeting LLMs: Towards Retrieval-Augmented Large Language Models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6491–6501.

[8] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997* (2023).

[9] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*. PMLR, 3929–3938.

[10] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[12] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. [n. d.]. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

[13] Gautier Izacard and Édouard Grave. 2021. 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*. 874–880.

[14] Gautier Izacard, Patrick Lewis, et al. 2022. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299* 1, 2 (2022), 4.

[15] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. 2020. A survey on contrastive self-supervised learning. *Technologies* 9, 1 (2020), 2.

[16] Joel Jang, Seungone Kim, et al. 2023. Personalized soups: Personalized large language model alignment via post-hoc parameter merging. *arXiv preprint arXiv:2310.11564* (2023).

[17] Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *International Conference on Machine Learning*. PMLR, 15696–15707.

[18] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[19] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[20] Patrick Lewis, Ethan Perez, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.

[21] Cheng Li, Mingyang Zhang, Qiaozhu Mei, Yaqing Wang, Spurthi Amba Hombaiah, Yi Liang, and Michael Bendersky. 2023. Teach LLMs to Personalize–An Approach inspired by Writing Education. *arXiv preprint arXiv:2308.07968* (2023).

[22] Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2024. Pre-trained language models for text generation: A survey. *Comput. Surveys* 56, 9 (2024), 1–39.

[23] Xinyu Li, Zachary C Lipton, and Liu Leqi. 2024. Personalized language modeling from personalized human feedback. *arXiv preprint arXiv:2402.05133* (2024).

[24] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.

[25] Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, et al. [n. d.]. RA-DIT: Retrieval-Augmented Dual Instruction Tuning. In *The Twelfth International Conference on Learning Representations*.

[26] Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[27] Sheshera Mysore, Zhuoran Lu, et al. 2023. Pearl: Personalizing large language model writing assistants with generation-calibrated retrievers. *arXiv preprint arXiv:2311.09180* (2023).

[28] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[29] Chris Richardson, Yao Zhang, Kellen Gillespie, Sudipta Kar, Arshdeep Singh, Zeynab Raeesy, Omar Zia Khan, and Abhinav Sethy. 2023. Integrating summarization and retrieval for enhanced personalization via large language models. *arXiv preprint arXiv:2310.20081* (2023).

[30] Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at TREC-3. *Nist Special Publication Sp* 109 (1995), 109.

[31] Alireza Salemi, Surya Kallumadi, and Hamed Zamani. 2024. Optimization methods for personalizing large language models through retrieval augmentation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 752–762.

[32] Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. 2023. Lamp: When large language models meet personalization. *arXiv preprint arXiv:2304.11406* (2023).

[33] Chenglei Shen, Xiao Zhang, Teng Shi, Changshuo Zhang, Guofu Xie, and Jun Xu. 2024. A survey of controllable learning: Methods and applications in information retrieval. *arXiv preprint arXiv:2407.06083* (2024).

[34] Teng Shi, Zihua Si, Jun Xu, Xiao Zhang, Xiaoxue Zang, Kai Zheng, Dewei Leng, Yanan Niu, and Yang Song. 2024. UniSAR: Modeling User Transition Behaviors between Search and Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1029–1039.

[35] Teng Shi, Jun Xu, Xiao Zhang, Xiaoxue Zang, Kai Zheng, Yang Song, and Enyun Yu. 2025. Unified Generative Search and Recommendation. arXiv:2504.05730 [cs.IR] https://arxiv.org/abs/2504.05730

[36] Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. REPLUG: Retrieval-Augmented Black-Box Language Models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 8364–8377.

[37] Zhaoxuan Tan, Zheyuan Liu, and Meng Jiang. 2024. Personalized Pieces: Efficient Personalized Large Language Models through Collaborative Efforts. *arXiv preprint arXiv:2406.10471* (2024).

[38] Zhaoxuan Tan, Qingkai Zeng, Yijun Tian, Zheyuan Liu, Bing Yin, and Meng Jiang. 2024. Democratizing Large Language Models via Personalized Parameter-Efficient Fine-tuning. arXiv:2402.04401 [cs.CL] https://arxiv.org/abs/2402.04401

[39] Jiakai Tang, Sunhao Dai, Teng Shi, Jun Xu, Xu Chen, Wen Chen, Wu Jian, and Yuning Jiang. 2025. Think Before Recommend: Unleashing the Latent Reasoning Power for Sequential Recommendation. arXiv:2503.22675 [cs.IR] https://arxiv.org/abs/2503.22675

[40] A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).

[41] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[42] Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. 2024. A survey on large language models for recommendation. *World Wide Web* 27, 5 (2024), 60.

[43] Xinghao Wu, Xuefeng Liu, Jianwei Niu, Haolin Wang, Shaojie Tang, and Guogang Zhu. 2024. FedLoRA: When Personalized Federated Learning Meets Low-Rank Adaptation. (2024).

[44] Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. 2024. Fine-grained human feedback gives better rewards for language model training. *Advances in Neural Information Processing Systems* 36 (2024).

[45] Zhuofeng Wu, Sinong Wang, Jiatao Gu, Madian Khabsa, Fei Sun, and Hao Ma. 2020. Clear: Contrastive learning for sentence representation. *arXiv preprint arXiv:2012.15466* (2020).

[46] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-Pack: Packaged Resources To Advance General Chinese Embedding. arXiv:2309.07597 [cs.CL]

[47] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. 2017. Deep matrix factorization models for recommender systems.. In *IJCAI*, Vol. 17. Melbourne, Australia, 3203–3209.

[48] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671* (2024).

[49] Changshuo Zhang, Teng Shi, Xiao Zhang, Qi Liu, Ruobing Xie, Jun Xu, and Ji-Rong Wen. 2024. Modeling Domain and Feedback Transitions for Cross-Domain Sequential Recommendation. *arXiv preprint arXiv:2408.08209* (2024).

[50] Changshuo Zhang, Teng Shi, Xiao Zhang, Yanping Zheng, Ruobing Xie, Qi Liu, Jun Xu, and Ji-Rong Wen. 2024. QAGCF: Graph Collaborative Filtering for Q&A Recommendation. *arXiv preprint arXiv:2406.04828* (2024).

[51] Changshuo Zhang, Xiao Zhang, Teng Shi, Jun Xu, and Ji-Rong Wen. 2025. Test-Time Alignment for Tracking User Interest Shifts in Sequential Recommendation. arXiv:2504.01489 [cs.IR] https://arxiv.org/abs/2504.01489

[52] Kepu Zhang, Teng Shi, Sunhao Dai, Xiao Zhang, Yinfeng Li, Jing Lu, Xiaoxue Zang, Yang Song, and Jun Xu. 2024. SAQRec: Aligning Recommender Systems to User Satisfaction via Questionnaire Feedback. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 3165–3175.

[53] Xiao Zhang, Teng Shi, Jun Xu, Zhenhua Dong, and Ji-Rong Wen. 2024. Model-Agnostic Causal Embedding Learning for Counterfactually Group-Fair Recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2024).

[54] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. 2023. Siren's song in the AI ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219* (2023).

[55] Wayne Xin Zhao, Jing Liu, Ruiyang Ren, and Ji-Rong Wen. 2024. Dense text retrieval based on pretrained language models: A survey. *ACM Transactions on Information Systems* 42, 4 (2024), 1–60.

[56] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).

[57] Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Haonan Chen, Zhicheng Dou, and Ji-Rong Wen. 2023. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107* (2023).

[58] Yuchen Zhuang, Haotian Sun, Yue Yu, Qifan Wang, Chao Zhang, and Bo Dai. 2024. HYDRA: Model Factorization Framework for Black-Box LLM Personalization. *arXiv preprint arXiv:2406.02888* (2024).