International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

An Explainable Retrieval-Augmented Generation-Based Personalized Recommendation System

S. Surekha[1], K. Kanthi Purnima[2]

[1]Associate Professor, Department of CSE, UCEK (A), JNTU Kakinada, Andhra Pradesh, India
[2]Student, Master of Computer Applications, UCEK (A), JNTU Kakinada, Andhra Pradesh, India

Abstract
Recommendation engines are significant tools that provide personalized recommendations especially in case of cold start problem where historical interaction data is insufficient. Deep learning techniques such as the state-of-the-art RAG (Retrieval-Augmented Generation) method make the recommendation procedure better by utilizing the concept of contextual preference. They, however, depend on large language models and therefore it increases the computational cost, slow the process and make it less explainable. This paper discusses a lightweight and explainable RAG-based recommendation engine. In the proposed model, SBERT embeddings and FAISS indexing are used for effective semantic retrieval and hybrid ranking. Explainability component is included in the architecture to provide explanations using similarity score, rating, popularity, author similarity and publication era. Experiments are performed on Kaggle Book-Crossing dataset using ranking metrics such as Recall, NDCG and MRR at various K values, where K is the number of recommendations selected from the ranked list. As per the experimental results, the proposed framework achieves its highest MRR score of 0.1509 at K = 25.

Keywords: Recommender system, retrieval-augmented generation, Sentence-BERT, FAISS, explainability, cold-start, NDCG, MRR.

1. INTRODUCTION

Recommendation systems have emerged as an indispensable part of the modern digital world. Recommendation systems are used to make suggestions for relevant products, movies, books, and other services by taking into consideration the preferences and behavior of the user. Due to the fast-paced increase of content and e-commerce, recommendation systems become more personalized and effective. One of the problems which arise from the work of recommendation systems is the cold start problem. The cold start problem occurs when there is insufficient data on the past interactions of users and items. Classical collaborative filtering method (CF) does not solve this problem since this recommendation technique is based on the historical interaction of user and item [1], [2]. Deep learning and LLM-based methods enhanced the accuracy of predictions through contextual reasoning but required high computation power. [3]

Moreover, most of the existing recommendation systems fail to provide explanations of the provided results. While users receive the recommendations they cannot understand the rationale of the given recommendation. The inability to explain the results lowers user confidence in the system and thus the

adoption process slows down. Explainable artificial intelligence systems have gained popularity over recent years and recommendation systems are no exception [4]. Such a method that can be mentioned is Retrieval-Augmented Generation (RAG), an appealing solution based on the combination of dense retrieval and generation approaches for the sake of improved output quality [11]. However, in spite of its advantages, the vast majority of RAG recommender systems use the LLM inference in autoregressive fashion and therefore suffer from the problem of latency making them unable to operate in real-time mode [5]. This paper proposes a novel solution for creating a lightweight and explainable RAG-based book recommendation system. Namely, Sentence-BERT embedding and FAISS similarity indexing are used in order to avoid the usage of LLMs at runtime.

The rest of the paper will follow a similar structure. The works that have been done in the field of recommendation systems and RAGs will be described in section II. Section III will demonstrate the proposed methodology for building the knowledge base and design of the query pipeline. Experiment setup and results will be presented in section IV. Conclusion and future works will be discussed in section V.

2. RELATED WORK

Ahmed et al. [1] proposed a trust-aware collaborative filtering model, where time-dependent trust information was acquired from social networks. In the research carried out by these authors, the enhanced performance of recommendations owing to the use of trust relations has been shown, however, this technique is possible only when there are social network data, and new users without any historical data cannot be considered. A review of collaborative filtering techniques has been done by Sharma et al. [2], where the problems of data sparsity and scalability were stated.

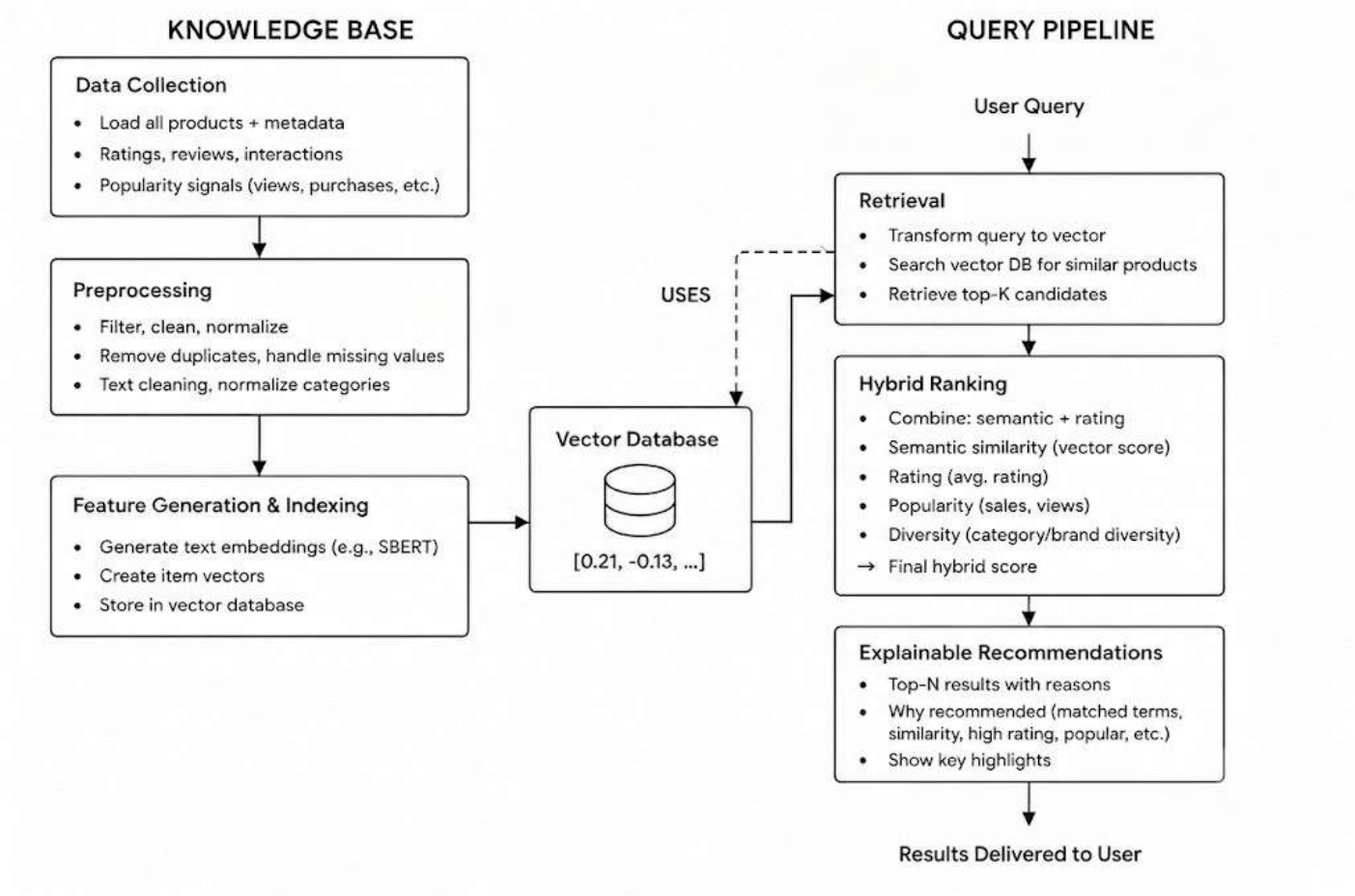
An overview of state-of-the-art recommendation techniques has been carried out by Adomavicius & Tuzhilin [3]. These authors have highlighted the problems of data sparsity and cold start, and even though the results obtained by them are rather old, they inspire the description of cold start problem in the recommended system which is based on semantic retrieval using content-based technique. The research of deep learning-based recommendation systems has been done by Zhang et al. [4], and the result shows that deep learning techniques are able to extract more latent features than hand-crafted ones.

It was proven by Bao et al. [5] that instruction tuning could generate recommendations matching users' preferences via LLM generation. Although this technique offers excellent performance, the scalability is limited due to the task-dependent autoregressive inference process used. Also, it was proven by Hou et al. [6] that LLMs can offer competitive zero-shot item ranking without performing finetuning. Nevertheless, even though the performance of the model is outstanding, the autoregressive generation has some inference constraints and fails to solve cold-start problems. Glass et al. observed that pre-generation document re-ranking helps in improving factual accuracy drastically. The retrieval and re-ranking approach suggested by the authors is similar to the hybrid approach of the proposed system using FAISS retrieval and re-scoring. Xu et al. studied the context window size for RAG tasks and observed that small k values negatively affect recall, while large values produce noise. This information is relevant to the hybrid ranking approach used to select top-k clean contexts.

3. METHODOLOGY

The proposed system is a lightweight and explainable recommendation framework designed for book recommendations using the Book-Crossing dataset. Fig. 1 provides an overall of the system architecture.

Figure 1: Workflow of the Proposed ERAG System



A. Data Collection and Preprocessing

First step involves a process of gathering and preprocessing of raw data for the semantic processing. The initial Book-Crossing dataset contains some missing values, dormant users and sparse interactions. Data preprocessing is carried out to improve data quality. The output from this step will be preprocessed dataset which will be used as input for feature generation and indexing.

1) Dataset collection: The Book-Crossing dataset is used in this research which is available on Kaggle. The Books.csv file has 270,491 records consisting of ISBN, title, author, publisher, publication year and image URL. While the Ratings.csv file has 1,149,766 records belonging to 105,274 users and 340,545 ISBNs having ratings ranging from 0 to 10 where 0 is implicit rating while 1-10 are explicit ratings. This dataset consists of approximately 99.6% sparsity and 38% explicit ratings.

2) Data preprocessing: Filtering process has been applied to increase the data quality and reduce sparsity by keeping only those books having at least 50 ratings and keeping at least 200 ratings of users. Missing values of title and author are removed; while publication year is converted to integers and renaming of columns is done.

B. Feature Generation and Indexing

After performing the preprocessing stage, the cleaned data from the books will be transformed into the semantic representation. Semantic embedding of each book will be created using the Sentence-BERT followed by indexing using the FAISS to build the knowledge base for semantic search.

1. Text to Vector Transformation: Every book will be represented using a complete sentence that contains details such as title, author name, publisher, and year of publishing the book. Fields with missing

information will be denoted using the empty string to prevent any occurrence of null values in the textual profile.

2. SBERT Embedding: In the proposed framework, the SBERT model is used to convert the book profile embeddings to 384 dimensional dense embedding vectors. The embedding of each vector is capable of capturing the semantic meaning of its respective profile. As a result, the comparison of the similarity can be performed using semantic meaning rather than keywords. The vectors are normalized using L2-normalization to make sure that the magnitude of each vector is the same. This step is important for consistency during the calculation of similarity using the inner product measure.
3. Vector Indexing and Storage (FAISS Index): The embedding vectors obtained from the models are indexed using FAISS IndexFlatIP. Fast nearest neighbor search using inner product similarity can be performed by using the vectors. These vector indexed embeddings are the retrieval component of the RAG pipeline. While recommending the books, semantically similar books obtained from the FAISS index provide more context for the ranking.

C. Retriever and Hybrid Ranking

Once the knowledge base is built, the system can start handling user queries. With the user's query input, the query is mapped to the same semantic space as that of the indexed book profiles. FAISS index returns the most relevant choices, and then they get re-ranked according to semantic similarity, rating, and popularity to generate personalized recommendations.

1. User Query Input: The user inputs the query in the form of free-text to indicate the type of book that he/she wishes to read. This query gets processed in the same manner as the book profiles were during the construction of the knowledge base.
2. Retrieval (FAISS-Based Semantic Search): The processed query gets embedded using the same SBERT model that generated the book profiles embeddings. These embeddings are searched through the FAISS index to retrieve the top 50 semantically similar books based on inner-product similarity. The retrieved candidates offer the contextual knowledge needed to make relevant recommendations with limited data or from scratch. The retrieval takes an average of 8.6 milliseconds.
3. Hybrid Ranking (Multi-Signal Re-ranking) : The retrieved candidates are further refined with a weighted hybrid scoring function shown in (1), Hybrid Score = 0.5 × Semantic Similarity + 0.3 × Average Rating + 0.2 × Popularity Score ...(1)

Where:

Semantic Similarity (50% weight): Inner-product similarity from FAISS retrieval on a scale of 0 to 1

Average Rating (30% weight): Mean user rating for the book on a scale of 0 to 1, normalized from 0 to 10

Popularity Score (20% weight): Normalized count of book ratings on a scale of 0 to 1

The Hybrid Ranker incorporates the semantic relevancy – the most significant signal, along with other signals from collaborative filtering, including the rating relevancy and popularity. The diversity filter ensures that at most two books by one author will be included in the generated list. The top five or ten books from the re-ranked list will be taken for the further processing stage.

Algorithm 1 : Explainable Retrieval-Augmented Recommendation Workflow

Input: Query Q, Dataset of Books D

Output: Top-N recommendation of books with explanations

/* Step : 1 Data collection & Preprocessing */

1. Load the Book-Crossing dataset.
2. Remove missing and duplicate records.
3. Filter users and books with insufficient interactions.
4. Create book profiles by combining title, author, publisher, and publication year.

/* Step : 2 Features generation & Indexing */

5. Generate the features for the book profiles by using Sentence-BERT.
6. Normalize the embedding vectors.
7. Build a FAISS IndexFlatIP using the normalized embeddings.

/* Step : 3 Retrieval & Hybrid ranking */

8. Accept the input query Q from the user.
9. Preprocess and generate the features for the query using Sentence-BERT.
10. Retrieve the Top-K books based on semantic similarity from the FAISS index.
11. Calculate the Hybrid Score of each retrieved book as:e

Hybrid Score = 0.5 x Semantic Similarity + 0.3 x Average Rating + 0.2 x Popularity Score

12. Rank books based on Hybrid Score in decreasing order.
13. Apply a diversity filter that allows a maximum of two books per author.
14. Get the Top-N ranked books.

/* Step : 4 Explainable recommendations */

15. Generate explanations using semantic similarity, average rating, popularity score, author information, and publication year.
16. Return the Top-N recommendations along with explanations.

D. Explainable Recommendations

Lastly, the top ranked recommendations are presented to the user along with the explanations for the same, which are generated using parameters like semantic similarity, user ratings, popularity, year of publication, and authorship of the books.

For each of the top five recommended books, three to five simple explanations are generated using the following five signals:

1. semantic similarity score – percentage similarity of the book with the user's query
2. reader rating – rating of the book by the users
3. publication year – year of publication of the book
4. popularity rank – ranking of the number of people who have rated the book
5. author match – the book is written by a popular or trending author

Such clearly articulated explanations help build the trust of the user through transparent reasoning. In contrast to the black box recommendation system, this recommendation system gives reasons for recom-

mending each of the books to the user.

E. Performance Evaluation Metrics

Evaluation is done based on IR metrics, which are calculated with various values of K (K = 5, 10, 15, 20, 25). Leave-second-half-out technique is used, where the number of selected users equals to 200. For each particular user, the first half of books that the user likes corresponds to the query and the second half of them is used as the ground truth.

1. Recall: Measures the proportion of relevant items successfully retrieved in the top-K list, as defined in (2), Recall = $\frac{|\text{Relevant} \cap \text{Retrieved@K}|}{|\text{Relevant}|}$ …. (2)
2. NDCG (Normalized Discounted Cumulative Gain): Evaluates ranking quality by rewarding relevant items appearing higher in the list, as defined in (3), NDCG = $\frac{\text{DCG@K}}{\text{IDCG@K}}$ …. (3)
3. MRR (Mean Reciprocal Rank): Computes the average reciprocal rank of the first relevant item in the top-K list, as defined in (4), MRR= $\left(\frac{1}{|U|}\right) \times \Sigma \left(\frac{1}{\text{rank}_i}\right)$ …. (4)

4. RESULTS AND ANALYSIS

A. Experimental Setup

The proposed approach was evaluated based on the Book-Crossing dataset provided by Kaggle. It consists of 270,491 books, 1,149,766 user ratings, and 105,274 users for 340,545 ISBNs. In order to remove noisy data and reduce sparsity, all the books with less than 50 user ratings and all users with less than 200 ratings were filtered out from the original dataset. As a consequence, the modified dataset with 1,934 books and 79 users was formed. The code was written in Python 3.10 language within Google Colab environment. All experiments were run on NVIDIA Tesla T4 GPU with 15 GB GPU memory and 12 GB of RAM. Semantic representations were generated using the Sentence-BERT model (all-MiniLM-L6-v2) in 384D space, and similarities retrieval was done with FAISS IndexFlatIP.

B. Evaluation Results

Table 1: Evaluation Metrics of the Proposed ERag System

K	Recall	NDCG	MRR
5	0.0456	0.0553	0.1287
10	0.0489	0.0539	0.1396
15	0.0460	0.0505	0.1456
20	0.0512	0.0530	0.1499
25	0.0529	0.0536	0.1509

Evaluation measures of the proposed system are tabulated below for all values of K in Table 1. From the results, it is evident that MRR increases progressively with an increase in K, ranging from 0.1287 when K=5 to 0.1509 when K=25. The result shows that the hybrid re-ranking system can recommend the relevant books. The maximum value in MRR is achieved for K=25, and its value is 0.1509. The result demonstrates excellent ranking and semantic retrieval in this case. Recall is also consistent for all values of K ranging between 0.0456 and 0.0529. It indicates a constant performance in the retrieval process. The

NDCG measure demonstrates good semantic ranking with a range between 0.0553 at K=5 up to 0.0505 at K=15, and then it is recovered to 0.0530-0.0536 at K=20-25.

C. Cold-Start Robustness Analysis

In the cold start robustness experiment, the recommendation accuracy of the proposed ERAG algorithm with regard to cold start users, where n <10 (n = 8), is evaluated for different values of K= 5, 10, 15, 20 and 25. The evaluation criterion selected in this context is Recall@K, because it assesses the percentage of relevant items among top K recommendations.

Table 2: Baseline Vs. ERAG (Cold-Start Users)

K	Baseline	ERAG
5	0.0500	0.0500
10	0.0500	0.0750
15	0.0750	0.0750
20	0.0750	0.1062
25	0.0750	0.1479

Recall@K metrics of the baseline and proposed RAG systems for cold-start users are provided in Table 2. As one can see, the proposed RAG model outperforms the baseline in all possible values of K. Namely, Recall@K increases from 0.0500 for K=5 to 0.1479 for K=25. Therefore, the SBERT-based algorithm of semantic retrieval identifies more relevant books as K grows. However, Recall@K for the baseline does not change and stays at the level of 0.0750 starting from K=15. Consequently, it implies that there is no way of finding more relevant books using the baseline algorithm which uses TF-IDF and popularity methods. Embeddings for context in the proposed RAG system allow for getting better results when dealing with sparse data. As seen from Table 2, the proposed RAG system demonstrates 41.6% higher results compared to the baseline for K=20 (0.1062 vs. 0.0750), and around 97% higher for K=25 (0.1479 vs. 0.0750).

It should be noted that the evaluation was conducted using a relatively small number of cold-start users (n = 8). Therefore, although Table 2 demonstrates the improved performance of the proposed system across all K values, the results are preliminary and require further validation. Future work will include evaluation using a larger cold-start user group.

D. Computational Efficiency

As to computational efficiency, the RAG model shows an average response time of 8.6 milliseconds (σ=1.1 ms) per request, which is suitable for interaction with the recommender application. Although the TF-IDF algorithm works faster with 2.4 milliseconds, the response time of 8.6 ms for the RAG model is considerably lower than 100 milliseconds needed for interactive systems. The low latency of 8.6 ms is justified by high-quality ranking with improvements in MRR by 102% compared to TF-IDF. The use of FAISS IndexFlatIP provides efficient vector search in 1,934 embeddings.

E. Baseline Comparison

To prove the efficiency of the proposed model, we compare the RAG system to three baselines – TF-IDF (text retrieval based on keywords), Popularity (the model with the highest average ratings) and Random (statistical baseline). With K=10, RAG outperforms the TF-IDF model by 35% (0.0489 vs. 0.0362) in

Recall metrics, by 73% (0.0539 vs. 0.0311) in NDCG metrics and by 102% (0.1396 vs. 0.0691) in MRR metrics. This proves that semantic embeddings are more effective than traditional keyword-based retrieval. Against the Popularity baseline, RAG shows higher ranking quality, outperforming it by 73% in NDCG and by 86% in MRR.

5. CONCLUSION

This study presented a lightweight and interpretable Retrieval-Augmented Generation model for personalized book recommendation, addressing large language model usage, the cold-start problem, and lack of interpretability. Using SBERT embeddings and FAISS-based similarity search, the model achieves fast semantic retrieval without compromising recommendation performance. The hybrid ranking algorithm (50% semantic, 30% rating, and 20% popularity) improves ranking quality, with MRR increasing from 0.1287 (K=5) to 0.1509 (K=25).

The implemented model outperforms baseline methods, demonstrating the efficiency of semantic search and hybrid ranking while remaining computationally efficient and effective in sparse-data recommendation tasks. For cold-start users, Recall@K improves from 0.0500 to 0.1479 (K=5 to K=25), outperforming the baseline by 41.6% at K=20 and 97.2% at K=25. However, due to the limited cold-start sample size (n=8), these results are indicative and require further validation.

The explainability module provides feature justifications through semantic similarity, reader ratings, popularity, author information, and publication period. Overall, the lightweight RAG framework demonstrates that semantic search with hybrid ranking delivers efficient and faster recommendations than traditional language-model-based systems. Future work will focus on advanced re-ranking techniques and personalized weights.

REFERENCES

1. R. Ahmed et al., “Trust and time in social network-based recommendations,” in Proc. ACM Conf. Recommender Systems (RecSys), 2020.
2. L. Sharma et al., “A survey of collaborative filtering approaches and challenges,” Int. J. Comput. Sci., vol. 14, no. 2, pp. 51–61, 2017.
3. G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: A survey,” IEEE Trans. Knowl. Data Eng., vol. 17, no. 6, pp. 734–749, 2005.
4. S. Zhang et al., “Deep learning based recommender system: A survey and new perspectives,” ACM Comput. Surv., vol. 52, no. 1, pp. 1–38, 2019.
5. K. Bao et al., “TALLRec: An effective and efficient tuning framework to align large language model with recommendation,” in Proc. ACM Conf. Recommender Systems (RecSys), 2023.
6. M. Hou et al., “Large language models are zero-shot rankers for recommender systems,” in Proc. Eur. Conf. Inf. Retrieval (ECIR), 2024.
7. M. Glass et al., “Re2G: Retrieve, rerank, generate,” in Proc. NAACL, 2022.
8. Y. Xu et al., “Retrieval-augmented generation with optimal context selection,” in Proc. AAAI Conf. Artif. Intell., 2024.
9. N. Reimers and I. Gurevych, “Sentence-BERT: Sentence embeddings using Siamese BERT-networks,” in Proc. EMNLP, 2019. [Online]. Available: https://arxiv.org/abs/1908.10084
10. J. Johnson, M. Douze, and H. Jégou, “Billion-scale similarity search with GPUs,” IEEE Trans. Big Data, vol. 7, no. 3, 2021. [Online]. Available: https://arxiv.org/abs/1702.08734

11. P. Lewis et al., “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in Proc. NeurIPS, 2020. [Online]. Available: https://arxiv.org/abs/2005.11401
12. Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” Computer, vol. 42, no. 8, pp. 30–37, 2009.
13. K. Järvelin and J. Kekäläinen, “Cumulated gain-based evaluation of IR techniques,” ACM Trans. Inf. Syst., vol. 20, no. 4, pp. 422–446, 2002.
14. C.-N. Ziegler et al., “Improving recommendation lists through topic diversification,” in Proc. WWW, 2005.
15. Arashnic, “Book recommendation dataset,” Kaggle, 2021. [Online]. Available: https://www.kaggle.com/datasets/arashnic/book-recommendation-dataset.