PUV: Personalized User Vectors Using Vector Database Recommender System

Dat Ngo-Thanh Nguyen*, Thang Phuoc Nguyen†, Tin Van Huynh‡
*†‡Faculty of Information Science and Engineering
University of Information Technology
Ho Chi Minh City, Vietnam
Emails: *21522923@gm.uit.edu.vn, †21522590@gm.uit.edu.vn, ‡tinhvh@uit.edu.vn

***Abstract*—With the rapid development of online platforms, recommendation systems have emerged to quickly provide users with the necessary information. Building personalized user profiles is crucial for accurate suggestions. However, creating user vectors through user demographics like age, location, work, etc., has many limitations due to users' unwillingness to provide information or providing incorrect information. In this paper, we propose PUV: Personalized User Vectors, a technique that addresses these shortcomings by personalizing user vectors based on the hotels they have reviewed. Additionally, the PipeCone vector database is used to store user vectors, improving accuracy when finding top-k similar users by utilizing the advantages of search algorithms. The best result we achieved was an RMSE of 1.581 using CafeBERT pre-trained word embedding description of the hotel. Our code is available at Github[1].**

***Index Terms*—Recommender System, Collaborative Filtering, User Vectors, Vector Database**

I. Introduction

Recommender systems play an important role in today's society due to the rapid development of online platforms. Providing information quickly to users, a good recommendation system not only helps improve user experience but also optimizes profits for the manufacturer. Building user vectors in recommender systems is one of the ways to increase the accuracy of the system, the purpose is to use vectors to accurately suggest items to users. Matrix factorization [1] presents techniques for decomposing the interaction matrix between users and items, such as Singular Value Decomposition (SVD) and Alternating Least Squares (ALS), from which user vectors are formed through hidden factors through matrix factorization. NCF [2] combines matrix factorization and neural networks to learn hidden vectors representing users and items, user vectors are learned through deep neural networks, thereby capturing hidden information about the relationship between users and items. In addition to models using interaction matrices between users and items, [3] [4] learns and predicts user preferences based on demographic characteristics, helping the system suggest suitable products or services. However, obtaining user information has many limitations, such as users not voluntarily providing information or providing false information, user information security is also very important in businesses such as banks, hospitals, etc. To solve the above problem, in this paper, we propose PUV (Personalized user vectors), which is a technique to personalize a user through the items that the user has used and the corresponding ratings. Each user has a different personality, so accurate personalization is necessary to improve the recommendation system. In addition, we use an impressive search engine through the search algorithms of Pinecone vectors database.

Our paper is organized as follows: first, section 2 discusses recent works using user vectors, PUV recommendation is presented in section 3. Then, the dataset and experiment results are presented in sections 4 and 5 respectively. Finally, this outline is summarized in section 6 conclusion, and future works.

[1]https://github.com/Sonny-Inkai/PERSONALIZED_USER_VECTOR_RECOMMENDER_SYSTEM

II. Related Works

A. Collaborative Filtering based on user vectors

Collaborative filtering is a long-standing method in recommendation systems, which is used to make product recommendations based on the opinion trends of users [5]. It can be divided into two topics, the first is Item-based collaborative filtering, Badrul Sarwa et al [6] introduced a recommendation method based on the items that users have used before, by using similarities measures such as correlation or cosine similarity between item vectors, this approach has gradually developed from the traditional method to the nonlinear neural network DeepICF [7], or using a hybrid method by combining Graph Neural Network and BERT Embeddings introduced by Javaji et al [8]. The second is user-based Colaborative Filtering, which is used to recommend items that are highly rated by people, or clusters that are most similar to the user introduced by Zhao et al [9] and Darban et al [4], which can use user demographics to create user vectors, from which users with high similarity can be searched. Both of these methods aim to personalize vectors for each user, to serve different purposes.

III. Proposed architecture

A. Personalized User Vectors

In this section, we focus on our proposed method which can be used to create user vectors. First, we define the notations used throughout the paper. Given a set of n users, $U = \{u_1, u_2, \ldots, u_n\}$, and a set of m items, $I = \{i_1, i_2, \ldots, i_m\}$, the interactions between users and items are represented by a

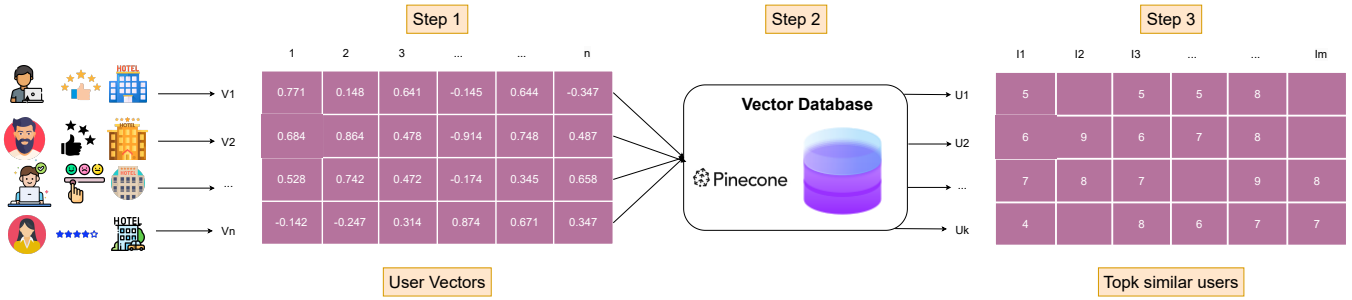


Fig. 1: The architecture of the proposed technique to users to vectors through their Hotel and its rating. Load all user vectors into Pinecone vector database to use its search algorithm to get top_k similar users and their ratings.

matrix R_{nm}, where r_{ui} is used to represent the interactions between item i and user u, r_{ui} = None if there is no interaction between user u and item i. $V = \{v_1, v_2, \ldots, v_n\}$ represent the vectors of users 1 to n, respectively. Second, to personalize the corresponding user vectors, we use pre-trained word embeddings, which are discussed in section 5, to convert the item descriptions into n-dimensional vectors, where n depends on the different pre-trained word embeddings, after that, we get $E = \{v_1, v_2, \ldots, v_m\}$ corresponding to the number of items. Next, multiply the corresponding r_{ui} by each word embedding v_i of the corresponding items, add them all up, and divide by the total r_{ui}, as in:

$$V_u = \frac{\sum_{i=0}^{m} E_i \cdot R_{ui}}{\sum_{i=0}^{m} R_{ui}} \quad (1)$$

As shown in Equation (1), if user u rates more items, V_u will be higher than other users, from this we have personalized vectors for each user.

B. Pinecone Vector Database

Pinecone provides long-term memory for high-performance AI applications. It's a managed, cloud-native vector database with a streamlined API and no infrastructure hassles. Pinecone serves fresh, relevant query results with low latency at the scale of billions of vectors. We leverage the fast and accurate capabilities of Pinecone to search for similar vectors using advanced search algorithms such as Approximate Nearest Neighbor (ANN) and Hierarchical Navigable Small World (HNSW) [10].

In our implementation, we focus on the efficiency and precision of these search algorithms to handle large-scale data. Approximate Nearest Neighbor (ANN) search allows us to quickly identify vectors that are close to a given query vector, reducing the computational load compared to exact search methods. The Hierarchical Navigable Small World (HNSW) algorithm enhances this by organizing vectors into a multi-layered graph structure, enabling rapid traversal and retrieval of nearest neighbors with high accuracy. This combination ensures that our system can meet the demands of real-time AI applications, providing relevant results swiftly and effectively.

By utilizing Pinecone's infrastructure, we eliminate the complexities associated with managing vector databases, allowing us to focus on optimizing our AI models and delivering superior performance. The scalability of Pinecone ensures that as our dataset grows, the search performance remains robust and reliable, handling billions of vectors without degradation in speed or accuracy.

C. Community Rating

In some cases, a few popular items receive a lot of attention, while the majority of the remaining items only attract a small number of users (long-tail items). When using PUV to get top_k similar users, but if that item is a long-tail item and top_k similar users do not have a rating for that item, then we will not be able to predict the rating for that item. To solve this problem, we use the Community Rating technique. By using the word embedding of item V_i , compare and get top_n similar items. Then, take the average ratings of top_n items by users who have rated those items. In this way, we solve the rating of long-tail items through the ratings of similar items.

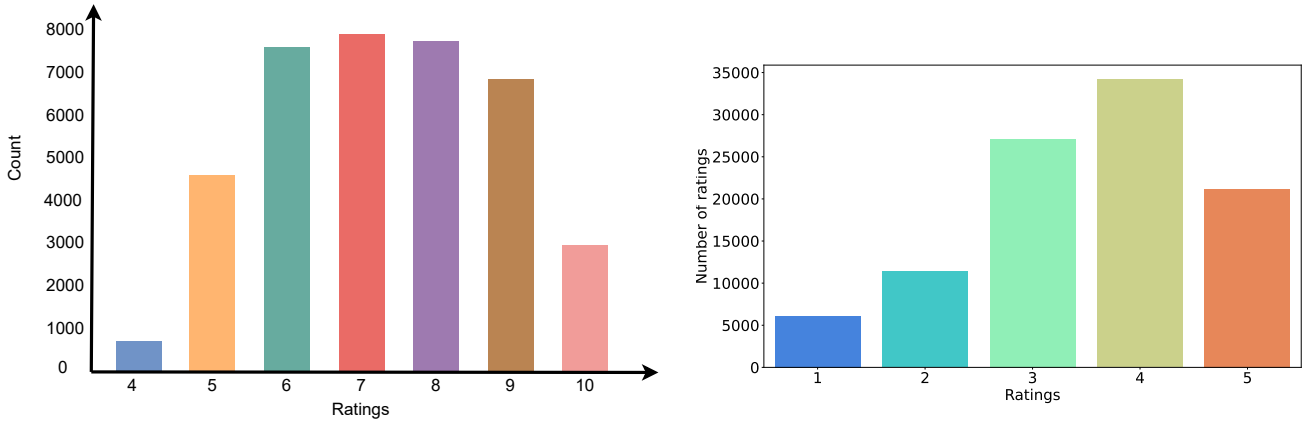
D. Overall Architechture

Fig. 1 depicts the architecture of our proposed Personalized User Vector (PUV), which is designed to enhance the recommendation process through several steps:

- In the initial step, the system personalizes user vectors. This involves capturing and processing various user-specific data points such as stay history and user ratings. By employing word embedding for hotel description, multiplied by its rating, we create a unique vector representation for each user. This personalized vector encapsulates the user's preferences, behaviors, and interactions within the platform, enabling a tailored recommendation experience.
- Step 2, once the user vectors are personalized, they are transferred to the Pinecone vector database. Pinecone is a managed, cloud-native vector database that offers a streamlined API and eliminates infrastructure hassles. It allows us to store and manage billions of vectors efficiently. The vectors uploaded to Pinecone are then indexed and made ready for rapid retrieval. This step ensures that the system can handle large-scale data and perform quick lookups to maintain low latency during recommendation queries.

Dataset	Users	Items	Ratings	Density
Hotel Booking DataSet	**6471**	**4506**	**38,801**	**0,133%**
MovieLens 100K	943	1,682	100,000	6,304%

TABLE I: The comparison table of data sparsity between Hotel Booking DataSet and MovieLens 100K DataSet



(a) Hotel Booking DataSet

(b) MovieLens 100K

Fig. 2: Illustration of rating ranges: (a) Hotel Booking DataSet captures a wider range with 10 rating labels compared to (b) MovieLens 100K, which only has 5 rating labels.

- In the third step, the system queries the Pinecone vector database to retrieve the top-k most similar users for a given user vector. This is achieved through sophisticated similarity search algorithms such as Approximate Nearest Neighbor (ANN) and Hierarchical Navigable Small World (HNSW) [10]. These algorithms ensure that the search is both fast and accurate, providing relevant results even at the scale of billions of vectors.
- Finally, from the pool of top-k similar users identified in the previous step, the system selects the top-m items with the highest ratings. These items are then recommended to the user. This final step leverages the collective wisdom of similar users to suggest items that are likely to be of high interest and value to the user. By focusing on highly rated items, the system aims to enhance user satisfaction and engagement with the platform.

IV. Dataset

A. Hotel Booking Rating Dataset

The dataset used in this study is published on the Kaggle website[2]. This dataset was collected by the author from the online booking website Booking.com, for the purpose of research on the topic. Details of the dataset including 38,801 rows of data, and 9 attributes are seen in Table II, with 4,506 hotels in 10 different provinces/cities, and 6,471 users.

B. Dataset Analysis

Ratings are rated on a scale of 0-10. The ratings in the dataset are distributed from 4-10 and are not evenly distributed, with the lowest rating being 4 and also the highest rating with the fewest number (nearly 1,000 votes), and the highest rating is 10 with nearly 3,000 votes. The rating range from 6-9 is quite even (number of votes is 7,000-8,000). We compare it with a famous dataset used in many recommender systems benchmarks, movielens 100k, the distribution of the two datasets is illustrated in Fig. 2. In Table I the density of the data is compared with the MovieLens 100K dataset.

[2]https://www.kaggle.com/datasets/phamtheds/hotel-booking-rating-dataset

Index	Features	Metadata
1	Hotel URL	Hotel URL
2	Location	Hotel location (Province/City name)
3	HotelID	Hotel ID
4	Hotel Name	Hotel Name
5	Descriptions	Hotel Description, Introduction
6	Address	Hotel address
7	UserID	User ID
8	User	User name (reviewer)
9	Ratings	User Rating

TABLE II: Details of Hotel Booking Dataset Features.

C. Data Pre-processing

To create a dataset for collaborative filtering based on users, we first count the number of ratings of each user to select users with 30 or more ratings, resulting in 139 users who satisfy the criteria. Next, we filter out those users from the Booking Hotels Dataset to create a new dataset.

From this dataset, for each different user, we choose 5 rows of data. This means that for each user, there will be 5 hotels that this user has rated to make the testing set. The training set is the remaining data in the Booking Hotels Dataset that has been removed from the data in the testing set. The results are two testing and training sets with 695 rows of data and 38,066 rows of data, respectively.

V. Empirical Evaluation

A. Baseline Models and Settings

To assess language comprehension, especially in Vietnamese, in this section we use word embeddings of 3 famous pre-trained models in Vietnam. First, PhoBERTv2 [11], trained exclusively on Vietnamese language data, exhibits high proficiency in understanding and processing Vietnamese text, allowing it to capture description information present in hotels, consequently enhancing the ac- curacy and effectiveness. The second is ViSoBERT [12] which is trained in Vietnamese and English data, boasts multilingual capabilities, enabling effective text processing in both languages. Its shared knowledge between the two languages during training enhance performance across diverse linguistic contexts. And finally, CafeBERT [13], trained on domain-specific data related to the culinary domain, possesses specialized knowledge in areas such as terminology and context. Its expertise in the culinary domain offers unique perspectives that complement analyses by PhoBERTv2 and ViSoBERT, enriching the overall capture of hotel description processing.

In addition, we also compare our PUV architecture with the three recommendation base-lines: GLocal-K [14] is a powerful method that combines global and local features to improve the performance of recommender systems. AutoRec [15] is an advanced method that combines autoencoder techniques with collaborative filtering to improve the accuracy of the recommender system. DeepICF [7] is a state-of-the-art model that uses deep neural networks to capture nonlinear and high-order relationships between items.

B. Evaluation Metric

In this study, we use the Root Mean Squared Error (RMSE) metric to evaluate and compare the experiments. RMSE is a popular metric in evaluating the performance of predictive models. It is calculated as the square root of the mean squared error between the predicted values and the actual values. The metric is illustrated in (2).

$$RMSE = \sqrt{\frac{\sum_{(ui)}(\hat{r}_{ui} - r_{ui})^2}{n}} \tag{2}$$

C. Experimental Results

Table III shows the performance of the baseline models on the test set. We compare three effective baseline models in the recommender system, GLocal-K [14], AutoRec [15], and DeepICF [7]. To assess the efficacy of our proposed PUV techinique, we incorporated different pre-trained word embeddings and tested each configuration. Our experiments demonstrate that the PUV model, when integrated with the CafeBERT pre-trained word embedding, outperforms all other baseline models. Specifically, the PUV model with CafeBERT achieved an RMSE of 1.581, marking it as the most effective configuration in our tests. Each model was subjected to five independent runs to ensure the reliability and robustness of our results. The average performance metrics from these runs are reported in Table III. The results clearly indicate that the choice of pre-trained word embeddings significantly influences the model's performance. Among the embeddings tested, CafeBERT consistently yielded superior results, thereby validating its effectiveness in enhancing the PUV model's predictive accuracy. Furthermore, our findings suggest that the integration of advanced pre-trained word embeddings can provide substantial improvements over traditional methods. This highlights the importance of leveraging state-of-the-art embeddings in the development of recommender systems. The comparative analysis underscores that our PUV architecture, augmented with CafeBERT, sets a new benchmark in the field, demonstrating its potential for practical applications in personalized recommendation scenarios. In summary, the experimental results underscore the benefits of using pre-trained word embeddings in enhancing the performance of recommender systems. The PUV model, particularly when coupled with CafeBERT, not only achieves the lowest RMSE but also exhibits consistent and reliable performance across multiple trials. This reinforces the value of our approach and its applicability in real-world settings.

Model	Hotel Booking Dataset
AutoRec [15]	2.145
DeepICF [7]	1.894
GLocal-K [14]	1.687
PUV$_{VisoBERT}$	1.609
PUV$_{PhoBERTv2}$	1.597
PUV$_{CafeBERT}$	**1.581**

TABLE III: The result of Hotel Booking Dataset on RMSE metric.

VI. Conclusion and Future Work

In this paper, we have presented our PUV (Personalized user vectors), which is a technique used to personalize users through the hotels that the user has rated, by using different pre-trained Vietnamese word embeddings to embed the hotel descriptions and using it together with its rating as a vector to represent the user.

In this way, we solve the problem of limiting the use of user demographics, in cases where users do not want to provide information or provide incorrect information. Besides, we use Pinecone vector database, which is used to store and search top_k similar user vectors quickly through ANN, and HNSW search algorithms. However, there is still a big problem which is cold start problems, when the user is a new member, we cannot create vectors for that user. To solve this problem, we propose a method to suggest trending products, thereby enriching information for users.

Acknowledgment

This work has been funded by The VNUHCM-University of Information Technology's Scientific Research Support Fund. We would like to thank anonymous reviewers from the International Conference on Knowledge and Systems Engineering 2024 (KSE 2024), which their comments helped us improve the quality of our paper.

REFERENCES

[1] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[2] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, pp. 173–182, 2017.

[3] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin, "Incorporating contextual information in recommender systems using a multidimensional approach," *ACM Transactions on Information systems (TOIS)*, vol. 23, no. 1, pp. 103–145, 2005.

[4] Z. Z. Darban and M. H. Valipour, "Ghrs: Graph-based hybrid recommendation system with application to movie recommendation," *Expert Systems with Applications*, vol. 200, p. 116850, 2022.

[5] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The adaptive web: methods and strategies of web personalization*, pp. 291–324, Springer, 2007.

[6] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, pp. 285–295, 2001.

[7] F. Xue, X. He, X. Wang, J. Xu, K. Liu, and R. Hong, "Deep item-based collaborative filtering for top-n recommendation," *ACM Transactions on Information Systems (TOIS)*, vol. 37, no. 3, pp. 1–25, 2019.

[8] S. R. Javaji and K. Sarode, "Hybrid recommendation system using graph neural network and bert embeddings," *arXiv preprint arXiv:2310.04878*, 2023.

[9] Z.-D. Zhao and M.-S. Shang, "User-based collaborative-filtering recommendation algorithms on hadoop," in *2010 third international conference on knowledge discovery and data mining*, pp. 478–481, IEEE, 2010.

[10] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 4, pp. 824–836, 2018.

[11] D. Q. Nguyen and A. T. Nguyen, "Phobert: Pre-trained language models for vietnamese," *arXiv preprint arXiv:2003.00744*, 2020.

[12] Q.-N. Nguyen, T. C. Phan, D.-V. Nguyen, and K. Van Nguyen, "Visobert: A pre-trained language model for vietnamese social media text processing," *arXiv preprint arXiv:2310.11166*, 2023.

[13] P. N.-T. Do, S. Q. Tran, P. G. Hoang, K. Van Nguyen, and N. L.-T. Nguyen, "Vlue: A new benchmark and multi-task knowledge transfer learning for vietnamese natural language understanding," *arXiv preprint arXiv:2403.15882*, 2024.

[14] S. C. Han, T. Lim, S. Long, B. Burgstaller, and J. Poon, "Glocal-k: Global and local kernels for recommender systems," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 3063–3067, 2021.

[15] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *Proceedings of the 24th international conference on World Wide Web*, pp. 111–112, 2015.