Ultra Fast Warm Start Solution for Graph Recommendations

Viacheslav Yusupov
HSE University
Moscow, Russian Federation
v.yusupov.lab@gmail.com

Maxim Rakhuba
HSE University
Moscow, Russian Federation

Evgeny Frolov
AIRI
Moscow, Russian Federation
HSE University
Moscow, Russian Federation

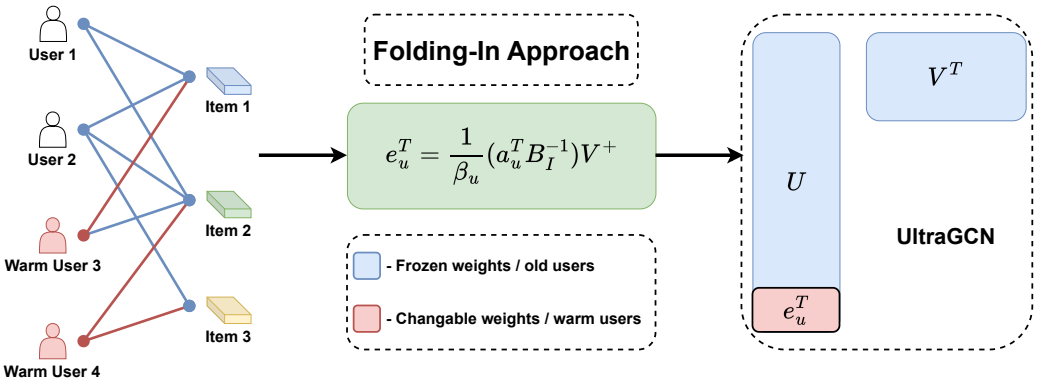

Figure 1: The image illustrates the warm-start scenarios, with warm users—those who have new interactions—marked in red, and others in blue. The scheme on the right presents the Folding-In approach for the UltraGCN [20] graph-based recommender system. The user embedding matrix is denoted as U, and the item embedding matrix as V. In the solution, a_u is the vector of user u interactions, the element β_u and the diagonal matrix B_I contain information about the graph structure and V^+ denotes a pseudo-inversed matrix. Trainable user embeddings (e_u) are shown in red, while frozen embeddings are shown in blue.

Abstract

In this work, we present a fast and effective Linear approach for updating recommendations in a scalable graph-based recommender system UltraGCN. Solving this task is extremely important to maintain the relevance of the recommendations under the conditions of a large amount of new data and changing user preferences. To address this issue, we adapt the simple yet effective low-rank approximation approach to the graph-based model. Our method delivers instantaneous recommendations that are up to 30 times faster than conventional methods, with gains in recommendation quality, and demonstrates high scalability even on the large catalogue datasets.

CCS Concepts

• **Computing methodologies** → **Machine learning algorithms**; • **Information systems** → **Personalization**; **Recommender systems**.

Keywords

Folding-In, Graph Neural Networks, Matrix Factorization, Collaborative Filtering, Recommender Systems, Scalability

ACM Reference Format:
Viacheslav Yusupov, Maxim Rakhuba, and Evgeny Frolov. 2025. Ultra Fast Warm Start Solution for Graph Recommendations. In *Proceedings of The 34th ACM International Conference on Information and Knowledge Management (CIKM '25).* ACM, New York, NY, USA, 5 pages. https://doi.org/https://doi.org/10.1145/3746252.3760932

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CIKM '25, Seoul, Korea
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/https://doi.org/10.1145/3746252.3760932

1 Introduction

Recommender systems play a crucial role in modern digital life; however, they face significant challenges due to a high information overload. To maintain the relevance of recommendations, it is essential that recommendation models are regularly updated. As the complexity of these models and the number of users continue to grow, the traditional model retraining becomes an inadequate solution for instant recommendations updates.

To address this issue, it is important to focus on updating recommendations only for those users who have interacted since the model was last trained or to tackle the *warm start* problem [10] using the *Folding-In* approach. Users with new interactions are referred to as *warm users* (see Figure 1). In recommender systems, sequential-based models [15, 22, 29] and graph-based models [32, 13, 20] have shown outstanding performance in various domains.

Recent advances in sequential-based models [31] have established a new state-of-the-art result in the field of sequential recommendations [22]. These models are fast and effective, effectively addressing the warm start problem by naturally updating user embeddings based on their interaction history [15]. However, sequential-based models may not yield significant benefits for certain datasets, particularly in cases where sequential patterns are absent [18].

In contrast, graph-based recommender systems [13, 20] learn the global information about user-item interactions according to the interaction graph. These models demonstrate high recommendation quality, capturing the complex dependencies in the data. However, graph approaches suffer from high computational demands and the necessity of updating user representations to maintain the reliability of recommendations [9, 36]. For this reason, it is crucial for graph-based models to develop Folding-In approaches to update user embeddings in the warm start scenario [14].

The existing Stochastic Gradient Descent-based Folding-In approaches [2] are universal but slow and computationally consuming. To overcome this problem, we developed a specialized Folding-In approach *Linear* for the simple and commonly used UltraGCN recommender system [20]. Compared to previous fine-tuning [34] and meta-learning [37] approaches, our method updates only one warm user embedding and could be applied on the fly with up to 30× speed-up and better quality of recommendations compared to conventional SGD or fine-tuning approaches. In addition, to demonstrate the effectiveness and linear scalability [26, 24] of our Linear method of updating recommendations, we tested it on datasets with large catalogues. Furthermore, our approach shows a higher diversity of recommendations due to the exact solution for updated embeddings, as well as the small impact on popularity bias [6].

Overall, our contributions are as follows.

- We developed a novel fast and efficient Folding-In approach Linear for graph-based recommender systems, achieving better quality than conventional SGD-based approaches with significantly 30× faster inference with better quality of recommendations.
- We theoretically and empirically study the scalability law of our Folding-In method.
- We demonstrate the effectiveness of our approach to increase the coverage of item catalogue.

The rest of the paper is organized as follows: Section 2 contains the related work, Section 3 provides an explanation of UltraGCN and SVD models, Section 4 details our approach, Sections 5, 6, and 7 contain our experimental setup, results, and conclusion, respectively. The code for reproducing our results is available[1].

2 Related Work

Graph-based recommender systems [32, 20, 13, 9] provide high-quality recommendations by leveraging the graph structure of the data and employing graph neural networks [33, 16]. Despite their effectiveness, training these systems often requires significant amounts of time and computational resources [36]. Consequently, retraining the model to capture new data becomes impractical in an online context. Some methods, such as conventional SGD-based approaches [2], are general but do not fully utilize the graph structure inherent in the model. In contrast, techniques such as metalearning [37] and graph prompting [34, 35, 30] update the entire model, which can be resource-intensive and time consuming. To address the limitations of these approaches, we have developed a fast and effective Folding-In approach for the UltraGCN model [20], which allows real-time updates of recommendations during inference and updates only several embeddings of users with a changed history of interactions.

[1] https://github.com/YusupovV-Lab/UltraFastFoldIn

3 Preliminaries

In this section, we provide some preliminaries of our approach and discuss the methods utilized in this work. In the following sections N denotes the number of items in the catalogue, M the number of users, and d is the size of the embeddings.

3.1 Folding-In for SVD

First, we start with a traditional matrix factorization approach [8], which solves the following problem:

$$\min_{B:\, rank(B)\leq d} \|A_0 - B\|_F^2 = \|A_0 - U\Sigma V^T\|_F^2 \tag{1}$$

where $A_0 \in \mathbb{R}^{M\times N}$ is the binary user-item interaction matrix and $U \in \mathbb{R}^{M\times d}$, $\Sigma \in \mathbb{R}^{d\times d}$ and $V \in \mathbb{R}^{N\times d}$ are the components of the SVD decomposition of A_0. The matrix U has the meaning of user embeddings and $V\Sigma$ is an item embedding matrix.

When A_0 changes rapidly through new interactions, model retraining is infeasible due to its computational cost. To overcome this problem, we can update only the embeddings of warm users who have new interactions[7, 8]. In this case, the optimization problem is formulated as follows:

$$\|a_u^T - e_u^T\Sigma V^T\|_2^2 \to \min_{e_u}, \tag{2}$$

where e_u is the embedding of the user u and the corresponding row of the matrix U. Despite the fact that this problem could be minimized by iterative approaches, such as SGD [2], the exact solution to the problem (2) be obtained explicitly using the orthogonality of V. The final solution for user u embedding has the $O(Nd)$ time and memory complexity:

$$e_u^T = (a_u^T V)\Sigma^{-1}. \tag{3}$$

Therefore, this folding-in approach provides fast and exact solutions without approximation errors and convergence issues as in optimization methods [2, 3].

3.2 Ultra Simplified Graph Neural Network

To test our Folding-IN approach, we utilize the efficient UltraGCN [20] model, which is the simplification of GCN-based recommender systems [13, 32]. In this model, the user-item interaction matrix is simplified to two vectors $b_U \in \mathbb{R}^M$ and $b_I \in \mathbb{R}^N$, collecting information about the total number of interactions for each user and item, respectively. Vectors b_U and b_I consist of elements β_u and β_i, where:

$$\beta_u = \frac{\sqrt{d_u + 1}}{d_u} \text{ and } \beta_i = \frac{1}{\sqrt{d_i + 1}}. \tag{4}$$

Additionally, we introduce the matrices $B_U = diag(b_U)$ and $B_I = diag(b_I)$ for shorter notation. The values d_u and d_i are the degrees

of vertices in the interaction graph, corresponding to the user u and the item i, respectively. The score function of the model is:

$$r_{ui} = \beta_u \beta_i e_u^T e_i, \tag{5}$$

where $e_u \in \mathbb{R}^d$ and $e_i \in \mathbb{R}^d$ are user and item embeddings. The model is trained with a loss function combined from the different BPR components [23] $\mathcal{L} = \mathcal{L}_B + \lambda \mathcal{L}_O$. In this expression:

$$\begin{aligned} \mathcal{L}_B &= -\sum_{u=1}^{M} \sum_{i \in \mathcal{N}_u} \sum_{j \notin \mathcal{N}_u} ln(\sigma(\beta_u \beta_i e_u^T e_i - \beta_u \beta_j e_u^T e_j)), \\ \mathcal{L}_O &= -\sum_{u=1}^{M} \sum_{i \in \mathcal{N}_u} \sum_{j \notin \mathcal{N}_u} ln(\sigma(e_u^T e_i - e_u^T e_j)), \end{aligned} \tag{6}$$

where $\mathcal{N}_u$ is a set of items with which the user u has interacted and $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function.

4 Addressing Warm Start

As could be seen in Subsection 3.2, the UltraGCN model [20] is similar to the matrix factorization model 3.1, where the user embedding matrix $B_U U$ in UltraGCN is similar to U in SVD, and $B_I V$ is similar to $V\Sigma$. Therefore, to update the user representations, we can utilize a similar optimization problem (2):

$$\|a_u^T - \beta_u e_u^T V^T B_I\|_2^2 \to \min_{e_u}. \tag{7}$$

Similarly to (2), we compute the derivate of (7) and set it equal to zero. The updated warm user embedding equal to:

$$e_u^T = \frac{1}{\beta_u}(a_u^T B_I^{-1}) V^+, \tag{8}$$

where B_I^{-1} is the diagonal matrix with $\frac{1}{\beta_i}$ on the diagonal and V^+ is the pseudo-inverse matrix $V^+ = A\Sigma_V^{-1}B^T$, where $V = A\Sigma_V B^T$ is the truncated SVD decomposition of the matrix V. We employ the pseudo-inverse of the initial matrix V in equation (2), as the orthogonality condition applicable in Singular Value Decomposition does not hold for matrix V [17].

Due to the fact that the item embeddings V do not change in our Folding-In procedure, the matrices V^+ and B_I^{-1} could be precomputed once and then reused for each embedding update. For this reason, the complexity of time and memory is equal to $O(Nd)$, where $c = a_u^T B_I^{-1}$ has $O(N)$ and $e_u^T = \frac{1}{\beta_u} c V^+$ has $O(Nd)$ complexity with only one matrix-vector multiplication. As a result, *our linear approach exhibits linear scalability with respect to catalogue size*. Moreover, due to the utilization of the history of only one user, the method has a small impact on popularity bias [1].

In contrast, conventional stochastic gradient descent-based approaches compute the gradient

$$\nabla \mathcal{L}_{e_u} = 2\beta_u V^T (B_I(\beta_u B_I(V e_u) - a_u)) \tag{9}$$

k times. Therefore, the complexity is $O(Ndk)$ with two matrix-vector multiplications for each gradient computation. Therefore, to improve the quality of the SGD-based method recommendations, different heuristics and interactions of other users are used. For instance, to improve the quality of recommendations, the final warm user embedding is computed as $e_u = \mu e_m + (1-\mu)e_u$, where $e_m = \frac{1}{M}\sum_{i=1}^{M} U_i$ is the mean user embedding, and $\mu \in [0, 1]$ is a hyperparameter of the method. This may have a negative effect on the scalability of the SGD-based approach due to additional computations and memory consumptions.

5 Experimental Setup

We evaluated our Folding-In approach against several others across four datasets with a wide range of catalogue sizes from $4 \cdot 10^3$ to $92 \cdot 10^3$: MovieLens-1M (ML-1M) [11], Amazon's Beauty and Books [12], and the Million Songs Dataset (MSD) [4]. The Books and MSD datasets are preprocessed similarly to work [27]. The statistics for these datasets are presented in Table 1. To ensure a robust evaluation and prevent information leakage [38], we split the datasets into three subsets (train, warm, and test) according to their timestamps [21, 5]. As a result, the datasets are divided into approximately 80%, 10%, and 10% proportions. The models are trained in the training subset, our methods are applied in the warm subset, and the performance is tested on the test subset.

In our comparisons, we compare the Folding-In approaches with the PureSVD [8] method, as well as the auto-encoder models: EASER [28] and SANSA [27]. These models do not utilize user embeddings; therefore, both models naturally solve the warm start problem. The SANSA is the sparse modification of the EASER model which has significantly faster training and inference time and utilizes notably less memory. We also compute the metrics of recommendations for the commonly used graph-based recommender system [13] which uses the simplification of graph convolution [16]. Additionally, we analyse various techniques for updating warm user representations in the UltraGCN model[20], including Zero (using initial model without Folding-In), Mean – the mean user embedding for each warm user, SGD-based approaches [2], Full model retraining, and our fast Folding-In method (Linear). Moreover, the Full retraining of the UltraGCN model [20] shows significantly faster training and inference with better scalability, compared to LightGCN [13] and EASER[28]. The performance of the models is evaluated using the Hit Rate and NDCG metrics [25].

Table 1: Dataset statistics.

Statistic	Users	Items	Actions	Density
ML-1M	6,040	3,706	1,000,209	5.43%
Beauty	22,363	12,101	198,502	0.073%
Books	632,458	91,599	35,918,135	0.062%
MSD	571,355	41,140	33,861,510	0.144%

6 Results

In this section, we present the results of our Linear Folding-In approach. The performance of our models is shown in Table 2. As demonstrated, our Folding-In approach significantly outperforms the SGD-based method in recommendation quality, achieving speeds up to 30× faster while providing superior recommendations. This improvement can be attributed to the use of the exact solution, as opposed to the approximations made by Stochastic Gradient Descent. Furthermore, due to the reduced number of hyperparameters compared to SGD-based approaches, identifying the optimal configuration for the method is considerably faster.

To illustrate the linear scalability of our method, we examine the relationship between the updating time per user and the catalogue size. In this experiment, we set the embedding size $d = 32$ for

Table 2: Performance comparison of different recommender approaches across datasets. Best metric values are highlighted in bold and second best values are underlined. H – HR metric, N – NDCG metric, sec/user – time in seconds of updating personal recommendations per user. Full retrain approach excluded from comparison due to inapplicable long training time.

Dataset	Metric	PureSVD	EASER	SANSA	LightGCN	Full	Zero	Mean	SGD	Linear (our)
ML-1M	H@5	0.0295	0.0324	0.0336	0.0354	0.0362	0.0341	0.0320	0.0354	**0.0357**
	H@10	0.0592	0.0605	0.0612	0.0678	0.0681	0.0625	0.0593	0.0667	**0.0675**
	N@5	0.0254	0.0266	0.0274	0.0288	0.0290	0.0283	0.0270	0.0285	**0.0287**
	N@10	0.0315	0.0341	0.0345	0.0370	0.0375	0.0357	0.0338	0.0367	**0.0369**
	sec/user	45.32	87.34	6.34	1629	729	–	–	0.223	**0.006**
Beauty	H@5	0.0230	0.0237	0.0227	0.0237	0.0241	0.0228	0.0221	0.0234	**0.0236**
	H@10	0.0436	0.0448	0.0425	0.0448	0.0453	0.0428	0.0421	0.0441	**0.0445**
	N@5	0.0158	0.0165	0.0164	0.0171	0.0172	0.0166	0.0159	0.0164	**0.0165**
	N@10	0.0247	0.0251	0.0243	0.0251	0.0254	0.0245	0.0240	0.0249	**0.0253**
	sec/user	113.56	205.86	13.45	2744	1175	–	–	0.872	**0.018**
Books	H@5	0.0265	0.283	0.0291	0.0316	0.0318	0.0301	0.0284	0.0307	**0.0311**
	H@10	0.0632	0.0684	0.0698	0.0782	0.0786	0.0748	0.0701	0.0766	**0.0774**
	N@5	0.0186	0.0193	0.0201	0.0216	0.0218	0.0209	0.0191	0.0212	**0.0215**
	N@10	0.0332	0.352	0.361	0.0391	0.0396	0.0379	0.0343	0.0386	**0.0390**
	sec/user	132.53	232.45	64.43	32465	11813	–	–	4.852	**0.145**
MSD	H@5	0.0273	0.305	0.0311	0.0331	0.0337	0.0326	0.0318	0.0329	**0.0334**
	H@10	0.0672	0.0703	0.0714	0.0775	0.0783	0.0759	0.0751	0.0765	**0.0772**
	N@5	0.0201	0.0214	0.0223	0.0242	0.0245	0.0231	0.0221	0.0235	**0.0240**
	N@10	0.0393	0.405	0.413	0.0427	0.0432	0.0413	0.0401	0.0421	**0.0426**
	sec/user	101.87	176.01	54.43	26746	9745	–	–	2.207	**0.067**

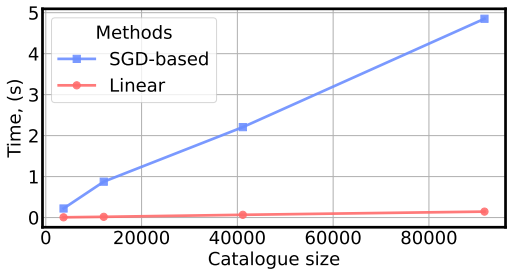

Figure 2: The comparison of time of updating warm user embedding for different catalogue sizes for SGD-based and our Linear approach. The points from the left to the right correspond to ML-1M, Beauty, MSD and Books datasets.

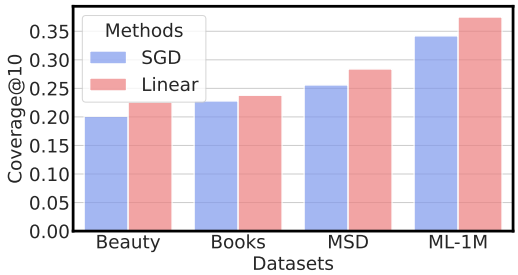

Figure 3: The comparison of coverage@10 on four datasets for the SGD-based and our Linear Folding-In approaches.

both the Linear and SGD-based approaches to empirically assess the scalability. As shown in Figure 2, our Linear approach exhibits linear scalability with a $O(N)$ complexity.

Moreover, by utilizing only the history of warm users, our Linear Folding-In approach effectively mitigates popularity bias [1]. In contrast, the tuned conventional SGD-based approach may use heuristics and the histories of other users (Section 4), which can adversely affect popularity bias in personalized recommendations. To investigate this effect, we compare the coverage metrics of the SGD-based and our Linear approaches. As shown in Figure 3, the coverage of SGD-based recommendations is notably lower than that of our Linear approach across all datasets. Therefore, the diversity of recommendations of our Linear method is also greater [1].

7 Conclusion

In this work, we developed a time- and resource-efficient scalable Folding-In approach to effectively address the warm start problem in UltraGCN-like recommender systems. Our Linear method achieves up to a 30× speedup over conventional SGD-based approaches while also enhancing the quality of personalized recommendations and demonstrating linear scalability with respect to catalogue size. Furthermore, our approach is less susceptible to popularity bias compared to traditional methods, resulting in more diverse recommendations.

Acknowledgments

Support from the Basic Research Program of HSE University is gratefully acknowledged. The calculations were performed in part through the computational resources of HPC facilities at HSE University [19].

References

[1] Himan Abdollahpouri. 2019. Popularity bias in ranking and recommendation. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, 529–530.

[2] Shun-ichi Amari. 1993. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5, 4-5, 185–196.

[3] Mehiddin Al-Baali, Emilio Spedicato, and Francesca Maggioni. 2014. Broyden's quasi-newton methods for a nonlinear system of equations and unconstrained optimization: a review and open problems. *Optimization Methods and Software*, 29, 5, 937–954.

[4] Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. 2011. The million song dataset. In *Ismir* number 9. Vol. 2, 10.

[5] Pedro G Campos, Fernando Díez, and Iván Cantador. 2014. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, 24, 67–119.

[6] Jiawei Chen, Hande Dong, Xiang Wang, Fuli Feng, Meng Wang, and Xiangnan He. 2023. Bias and debias in recommender system: a survey and future directions. *ACM Transactions on Information Systems*, 41, 3, 1–39.

[7] Michael D Ekstrand, John T Riedl, Joseph A Konstan, et al. 2011. Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction*, 4, 2, 81–173.

[8] G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum. 1988. Information retrieval using a singular value decomposition model of latent semantic structure. In *Proceedings of the 11th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (SIGIR '88). Association for Computing Machinery, Grenoble, France, 465–480. ISBN: 2706103094. doi:10.1145/62437.62487.

[9] Chen Gao et al. 2023. A survey of graph neural networks for recommender systems: challenges, methods, and directions. *ACM Transactions on Recommender Systems*, 1, 1, 1–51.

[10] Anupriya Gogna and Angshul Majumdar. 2015. A comprehensive recommender system model: improving accuracy for both warm and cold start users. *IEEE Access*, 3, 2803–2813.

[11] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: history and context. *Acm transactions on interactive intelligent systems (tiis)*, 5, 4, 1–19.

[12] Ruining He and Julian McAuley. 2016. Ups and downs: modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, 507–517.

[13] 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.

[14] Guillermo Jorge-Botana, Jose A Leon, Ricardo Olmos, and Inmaculada Escudero. 2010. Latent semantic analysis parameters for essay evaluation using small-scale corpora. *Journal of Quantitative Linguistics*, 17, 1, 1–29.

[15] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.

[16] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

[17] Virginia Klema and Alan Laub. 1980. The singular value decomposition: its computation and some applications. *IEEE Transactions on automatic control*, 25, 2, 164–176.

[18] Anton Klenitskiy, Anna Volodkevich, Anton Pembek, and Alexey Vasilev. 2024. Does it look sequential? an analysis of datasets for evaluation of sequential recommendations. In *Proceedings of the 18th ACM Conference on Recommender Systems*, 1067–1072.

[19] PS Kostenetskiy, RA Chulkevich, and VI Kozyrev. 2021. Hpc resources of the higher school of economics. In *Journal of Physics: Conference Series* number 1. Vol. 1740. IOP Publishing, 012050.

[20] Kelong Mao, Jieming Zhu, Xi Xiao, Biao Lu, Zhaowei Wang, and Xiuqiang He. 2021. Ultragcn: ultra simplification of graph convolutional networks for recommendation. In *Proceedings of the 30th ACM international conference on information & knowledge management*, 1253–1262.

[21] Zaiqiao Meng, Richard McCreadie, Craig Macdonald, and Iadh Ounis. 2020. Exploring data splitting strategies for the evaluation of recommendation models. In *Proceedings of the 14th acm conference on recommender systems*, 681–686.

[22] Gleb Mezentsev, Danil Gusak, Ivan Oseledets, and Evgeny Frolov. 2024. Scalable cross-entropy loss for sequential recommendations with large item catalogs. In *Proceedings of the 18th ACM Conference on Recommender Systems*, 475–485.

[23] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. Bpr: bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence* (UAI '09). AUAI Press, Montreal, Quebec, Canada, 452–461. ISBN: 9780974903958.

[24] Deepjyoti Roy and Mala Dutta. 2022. A systematic review and research perspective on recommender systems. *Journal of Big Data*, 9, 1, 59.

[25] Thiago Silveira, Min Zhang, Xiao Lin, Yiqun Liu, and Shaoping Ma. 2019. How good your recommender system is? a survey on evaluations in recommendation. *International Journal of Machine Learning and Cybernetics*, 10, 813–831.

[26] Monika Singh. 2020. Scalability and sparsity issues in recommender datasets: a survey. *Knowledge and Information Systems*, 62, 1, 1–43.

[27] Martin Spišák, Radek Bartyzal, Antonín Hoskovec, Ladislav Peska, and Miroslav Tůma. 2023. Scalable approximate nonsymmetric autoencoder for collaborative filtering. In *Proceedings of the 17th ACM Conference on Recommender Systems*, 763–770.

[28] Harald Steck. 2019. Embarrassingly shallow autoencoders for sparse data. In *The World Wide Web Conference*, 3251–3257.

[29] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. Bert4rec: sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, 1441–1450.

[30] Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. 2022. Gppt: graph pre-training and prompt tuning to generalize graph neural networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 1717–1727.

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

[32] 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.

[33] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32, 1, 4–24.

[34] Yuhao Yang, Lianghao Xia, Da Luo, Kangyi Lin, and Chao Huang. 2024. Graphpro: graph pre-training and prompt learning for recommendation. In *Proceedings of the ACM Web Conference 2024*, 3690–3699.

[35] Zixuan Yi, Iadh Ounis, and Craig Macdonald. 2023. Contrastive graph prompt-tuning for cross-domain recommendation. *ACM Transactions on Information Systems*, 42, 2, 1–28.

[36] Jiahao Zhang, Rui Xue, Wenqi Fan, Xin Xu, Qing Li, Jian Pei, and Xiaorui Liu. 2024. Linear-time graph neural networks for scalable recommendations. In *Proceedings of the ACM Web Conference 2024*, 3533–3544.

[37] Yang Zhang, Fuli Feng, Chenxu Wang, Xiangnan He, Meng Wang, Yan Li, and Yongdong Zhang. 2020. How to retrain recommender system? a sequential meta-learning method. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1479–1488.

[38] Jun-Jie Zhu, Meiqi Yang, and Zhiyong Jason Ren. 2023. Machine learning in environmental research: common pitfalls and best practices. *Environmental Science & Technology*, 57, 46, 17671–17689.

GenAI Usage Disclosure

In preparing this paper, we used DeepSeek and ChatGPT to identify and correct spelling errors, typos, and improve the clarity of the text. No AI tools were used for data analysis, experimentation, formulation of conclusions, and mathematical formulations. The research methodology, results, and all other technical aspects of the work were developed by the authors of the article without using GenAI. We utilize DeepSeek and ChatGPT only for proofreading and stylistic issues, while we maintain the full academic integrity and originality of our research.