Data Augmentation using Reverse Prompt for Cost-Efficient Cold-Start Recommendation

Genki Kusano
g-kusano@nec.com
NEC Corporation
Kawasaki, Japan

ABSTRACT

Recommendation systems that use auxiliary information such as product names and categories have been proposed to address the cold-start problem. However, these methods do not perform well when we only have insufficient warm-start training data. On the other hand, large language models (LLMs) can perform as effective cold-start recommendation systems even with limited warm-start data. However, they require numerous API calls for inferences, which leads to high operational costs in terms of time and money. This is a significant concern in industrial applications. In this paper, we introduce a new method, *RevAug*, which leverages LLMs as a data augmentation to enhance cost-efficient cold-start recommendation systems. To generate pseudo-samples, we have reversed the commonly used prompt for an LLM from "Would this user like this item?" to "What kind of items would this user like?". Generated outputs by this reverse prompt are pseudo-auxiliary information utilized to enhance recommendation systems in the training phase. In numerical experiments with four real-world datasets, RevAug demonstrated superior performance in cold-start settings with limited warm-start data compared to existing methods. Moreover, RevAug significantly reduced API fees and processing time compared to an LLM-based recommendation method.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

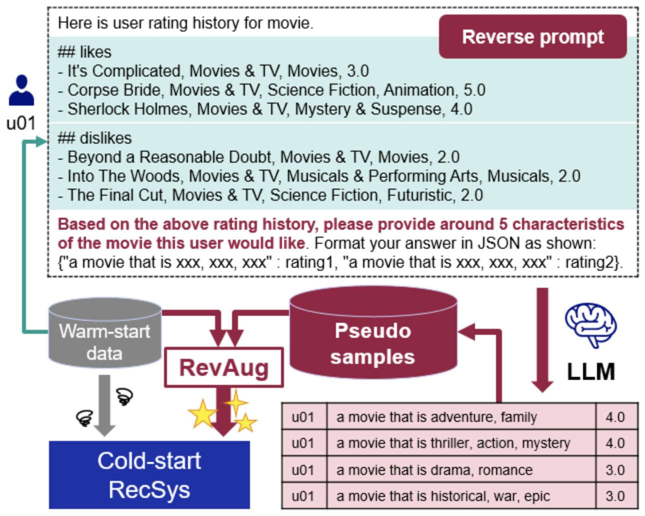
Cold-Start Recommendation, Data Augmentation, Large Language Model, Reverse Prompt

ACM Reference Format:
Genki Kusano. 2024. Data Augmentation using Reverse Prompt for Cost-Efficient Cold-Start Recommendation. In *18th ACM Conference on Recommender Systems (RecSys '24), October 14–18, 2024, Bari, Italy.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3640457.3688159

1 INTRODUCTION

Collaborative filtering techniques like matrix factorization [4, 8, 14] are well-known recommendation methods. They learn from user-item interactions within training data to predict user reactions to their unseen items. However, these methods suffer from recommending items not included in the training data, such as new products, which is known as the cold-start problem.

Recommendation systems [1, 17, 19, 23, 25] using auxiliary information such as product names and categories have been developed to resolve the cold-start problem. These methods learn not only collaborative embeddings for users and items within the training data but also latent features for the auxiliary information. Then, cold-start items can be represented as latent features from their auxiliary information, although their collaborative embeddings cannot be computed. Hence, it is enabled to predict user reactions for the cold-start items by the auxiliary latent features. However, training effective latent features requires training data of sufficient quantity and quality. Companies often do not have enough data, particularly right after the release of a new business project like a startup, which makes solving cold-start problems difficult.



Here is user rating history for movie.
likes
- It's Complicated, Movies & TV, Movies, 3.0
- Corpse Bride, Movies & TV, Science Fiction, Animation, 5.0
- Sherlock Holmes, Movies & TV, Mystery & Suspense, 4.0
dislikes
- Beyond a Reasonable Doubt, Movies & TV, Movies, 2.0
- Into The Woods, Movies & TV, Musicals & Performing Arts, Musicals, 2.0
- The Final Cut, Movies & TV, Science Fiction, Futuristic, 2.0

Based on the above rating history, please provide around 5 characteristics of the movie this user would like. Format your answer in JSON as shown: {"a movie that is xxx, xxx, xxx" : rating1, "a movie that is xxx, xxx, xxx" : rating2}.

u01	a movie that is adventure, family	4.0
u01	a movie that is thriller, action, mystery	4.0
u01	a movie that is drama, romance	3.0
u01	a movie that is historical, war, epic	3.0

Figure 1: Overview of the proposed method, RevAug. Conventional cold-start recommendation methods are trained only with warm-start data. RevAug creates prompts from this warm-start data, uses an LLM to generate pseudo-samples, and applies these to enhance the recommendation system.

Recent advancements in large language models (LLMs) give a new type of cold-start recommendation systems by utilizing their vast knowledge and inference skills, even without training data [6, 9, 10, 15, 20–22, 24]. However, these approaches suffer from the high costs required for inferences. For instance, predicting the

RecSys '24, October 14–18, 2024, Bari, Italy
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0505-2/24/10
https://doi.org/10.1145/3640457.3688159

reactions of 1000 users to 100 items leads to 100,000 API calls for an LLM, which leads to extensive processing times and API fees. While some studies [3, 16] aim to reduce inference costs by using LLMs during training rather than inference, they do not address the cold-start problem. If the details of cold-start items are known in advance, Wang et al. [18] proposed a cost-efficient recommendation system for these items. However, this system requires running LLMs and retraining the model whenever new cold-start items appear, making it inefficient to infer arbitrary cold-start items.

In this paper, we propose *RevAug*, a method that improves cold-start recommendation systems with insufficient training data while reducing inference costs. RevAug architecture generalizes Heater [25] and CLCRec [19], which are cost-efficient cold-start recommendation methods using auxiliary information. We address the issue of poor recommendation performance due to limited training data by using data augmentation with LLMs. Standard LLM-based recommendation systems [9, 10, 18, 21] ask the LLM, "Would this user like this item?", where this kind of question for LLMs is called a prompt. These researches suggest that LLMs contain adequate knowledge for recommendations. To extract this knowledge, we invert the former prompt to "What kind of items would this user like?" (Figure 1). With this reverse prompt[1], the LLM generates auxiliary information about items the user might like, which we use to train the recommendation model. Since the outputs are pseudo-samples, we adjusted the base architecture of Heater and CLCRec with a specific regularizer to improve recommendation performance.

To evaluate the effectiveness of RevAug, we conducted several numerical experiments using real-world data containing less than ten thousand user-item interactions. We confirmed that RevAug performed superior to other comparison methods in situations with limited training data. Furthermore, we verified that inference with RevAug significantly reduced processing time by up to 99.9% compared to an LLM-based recommendation system. These results validate the superiority of our proposed approach.

2 COLD-START RECOMMENDATION

Let U and I be the sets of users and items, respectively, and $\mathcal{R} \subset U \times I$ be the set of user-item interactions. For a user-item pair $(u, i) \in \mathcal{R}$, the feedback given by user u for item i is represented by $r_{u,i}$. This feedback takes real numbers for explicit feedback, such as ratings, or $\{0, 1\}$ for implicit feedback, such as whether the item was bought.

In this paper, we generalize Heater [25] and CLCRec [19] into a cold-start recommendation system that uses auxiliary information. We assume that auxiliary information for an item is represented as a text, such as titles or categories, and vectorize it to $\mathbf{x}_i \in \mathbb{R}^\ell$ through sentence embedding techniques such as SimCSE [5].

Table 1: Loss functions.

	Heater [25] (Explicit)	CLCRec [19] (Implicit)
$\ell_I(u, i)$	$\lvert r_{u,i} - \mathbf{f}_u^T \mathbf{f}_i \rvert^2$	$h(\mathbf{f}_u, \mathbf{f}_i, \{\mathbf{f}_{i_k}\}_{n=1}^N)$
$\ell_A(i)$	$\lVert \mathbf{W}\mathbf{x}_i - \mathbf{z}_i \rVert^2$	$h(\mathbf{V}\mathbf{z}_i, \mathbf{f}_i, \{\mathbf{f}_{i'_n}\}_{n=1}^N)$

Before explaining the details of Heater and CLCRec, let us prepare notations. Let $\mathbf{z}_u, \mathbf{z}_i \in \mathbb{R}^k$ be embeddings of u and i to be trained from their interactions, respectively. We consider that the sentence embedding $\mathbf{x}_i$ of item auxiliary information is transformed to be closer to $\mathbf{z}_i$ through the parameter matrix $\mathbf{W} \in \mathbb{R}^{k \times \ell}$ as $\mathbf{W}\mathbf{x}_i \approx \mathbf{z}_i$. Other parameter matrices $\mathbf{V}, \mathbf{V}' \in \mathbb{R}^{d \times k}$ are introduced, and latent embeddings are formulated as $\mathbf{f}_i := \mathbf{V}\mathbf{W}\mathbf{x}_i$ and $\mathbf{f}_u := \mathbf{V}'\mathbf{z}_u$. For $(u, i) \in \mathcal{R}$, let $i_1, \dots, i_N$ be randomly sampled items which have not been interacted by u, that is, $(u, i_n) \notin \mathcal{R}$. For $i \in I$, let $i'_1, \dots, i'_N$ be randomly sampled items that are not i. For $\mathbf{a}, \mathbf{b}, \mathbf{b}_1, \dots, \mathbf{b}_N \in \mathbb{R}^d$ and $\tau > 0$, we define a function h as follows:

$$h(\mathbf{a}, \mathbf{b}, \{\mathbf{b}_n\}_{n=1}^N) := -\log\left(\frac{\hat{p}(\mathbf{a}, \mathbf{b})}{\hat{p}(\mathbf{a}, \mathbf{b}) + \sum_{n=1}^N \hat{p}(\mathbf{a}, \mathbf{b}_n)}\right) \tag{1}$$

where $\hat{p}(\mathbf{a}, \mathbf{b}) := \exp(\tau^{-1}\langle \mathbf{a}, \mathbf{b}\rangle / \lVert \mathbf{a} \rVert \lVert \mathbf{b} \rVert)$. The function h originates from contrastive learning [2, 7], where it decreases h when $\mathbf{a}$ and $\mathbf{b}$ are similar, or when $\mathbf{a}$ is not similar to any $\mathbf{b}_n$.

With these symbols and Table 1, the loss functions of Heater and CLCRec are expressed as the sum of the loss ℓ_I related to user-item interactions and that ℓ_A related to auxiliary information[2][3]:

$$\mathcal{L}_{Base} := \mathbb{E}_{(u,i) \in \mathcal{R}}[\ell_I(u, i)] + \alpha \mathbb{E}_{i \in I}[\ell_A(i)] + \lambda \lVert \Theta \rVert_F^2 \tag{2}$$

where $\alpha > 0, \lambda > 0$ and $\Theta = (\mathbf{W}, \mathbf{V}, \mathbf{V}', \{\mathbf{z}_u\}_{u \in U}, \{\mathbf{z}_i\}_{i \in I})$. During optimizing Equation (2), we apply a technique to enhance learning efficiency by replacing $\mathbf{f}_i = \mathbf{V}\mathbf{W}\mathbf{x}_i$ with $\mathbf{V}\mathbf{z}_i$ at a probability $p \in [0, 1]$, which is known as a randomized training [25].

3 REVAUG

In this section, we introduce RevAug, a proposed method that generates pseudo-samples using LLMs from $\mathcal{R}$ and utilizes it for training in cold-start recommendation systems.

3.1 Reverse Prompt

As mentioned in the introduction, we utilize LLMs to augment training data to enhance the performance of cost-effective cold-start recommendations. Here, we select m favorite items from a user's history as few-shot samples to include in prompts. The criterion for "favorite" items is arbitrary. In cases of explicit feedback with ratings from 1 to 5, items rated three or higher are considered favorites in our experiments. For implicit feedback cases, we choose them from positive items of $r_{u,i} = 1$. Disliked items can also be included in the prompts. An example of the suggested prompt is shown in Figure 1 (Detailed prompts are provided in the appendix). Finally, we obtain tuples of (userID, item text, reaction) from one user. This collection of all users' pseudo-samples is denoted as $\mathcal{R}_{Aug}$.

3.2 Training

For a pseudo-sample $(u, i') \in \mathcal{R}_{Aug}$, the item text generated by an LLM can be converted into a sentence embedding $\mathbf{x}_{i'}$. This allows for the calculation of latent feature $\mathbf{f}_{i'} = \mathbf{V}\mathbf{W}\mathbf{x}_{i'}$ and interaction loss $\ell_I(u, i')$. Considering these are pseudo-samples, we introduce

[1] *RevAug* is named from reverse prompt for data augmentation.

[2] The transformation using $\mathbf{V}$ was mentioned in Heater but not introduced in CLCRec. However, by setting $\mathbf{V}$ as the identity matrix, it can reproduce CLCRec. We apply $\mathbf{V}$ to CLCRec as a generalized setting.

[3] Due to space constraints, we limit auxiliary information to items only. If a latent feature $\mathbf{x}_u$ exists for the user u, the loss $\ell_A(u)$ of user auxiliary information can be similarly defined.

Table 2: Dataset statistics.

	Movie	Music	Book	Grocery
user	3000	573	1591	570
interactions (train)	6212	1250	2829	1024
interactions (cold)	3096	823	841	426

a weight parameter $\beta > 0$ for pseudo-samples and define the loss function of RevAug as follows:

$$\mathcal{L}_{RevAug} := \mathcal{L}_{Base} + \beta \mathbb{E}_{(u,i') \in \mathcal{R}_{Aug}}[\ell_I(u, i')]. \tag{3}$$

Here, we explain how to set the hyper-parameter β. First, we divide a warm-start data $\mathcal{R}$ into a training set $\mathcal{R}_{tr}$ and a validation set $\mathcal{R}_{val}$. Then, we obtain parameters Θ_β by minimizing Equation (3) on $\mathcal{R}_{tr}$. After training, we compute the performance score on $\mathcal{R}_{val}$, denoted by score($\mathcal{R}_{val}; \Theta_\beta$), where it is the mean absolute error (MAE) and normalized discounted cumulative gain (nDCG) score for explicit and implicit feedback cases, respectively[4]. Finally, we adopt $\hat{\beta} = \text{argmin}_\beta$ score($\mathcal{R}_{val}; \Theta_\beta$) as the selected hyper-parameter.

4 EXPERIMENTS

To verify the effectiveness of RevAug, we conducted numerical experiments. The research questions (RQ) are as follows: **RQ1**: How well does RevAug perform cold-start recommendations compared to other methods when only limited training data is available? **RQ2**: What minimum training data is required for RevAug to perform effectively? **RQ3**: How much does RevAug reduce inference costs?

4.1 Settings

4.1.1 Dataset. We used four categories: Movie, Music, Book, and Grocery from Amazon Review Dataset[12][5]. This dataset includes user ratings ranging from 1 to 5, which we used as explicit feedback. In the context of implicit feedback, we treated all items with users' ratings as positive, that is, $r_{u,i} = 1$.

There are titles and categories as item auxiliary information, and we set an item text to "An item entitled with {title} has {categories} tags.". Since there is no user auxiliary information, user texts are not given in these experiments.

As an item set, we selected 100 items and treated them as warm-start items I. We fix users with interactions for items as U. Due to the large number of users in the Movie dataset, we limited this to 3000 users and selected randomly until this number was reached. Users in U had interactions with items beyond I. We randomly assigned 100 items from these and defined them as cold-start items. Table 2 presents the number of users and the number of interactions for training and cold-start items in each dataset.

4.1.2 Comparison methods. We compared RevAug with the following methods:

Baseline: It returns the average rating of all interactions in the training data for explicit feedback. For implicit feedback, top-K items are randomly sorted and used for calculating the nDCG@K score. Inspired by [4], we set these predictions as minimal baselines that can work in a cold-start setting without training data.

Base model: We use Heater for explicit feedback and CLCRec for implicit feedback. These are RevAug without data augmentation, equivalent to $\beta \equiv 0$ in Equation (3).

Ablation: To evaluate the effectiveness of LLM-generated data, we intentionally create an inaccurate augmented data $\mathcal{R}^{Ab}_{Aug}$ and use it for training as an ablation study. For explicit feedback, we set $\mathcal{R}^{Ab}_{Aug} := \{(u, i', 5 - r \mid (u, i', r) \in \mathcal{R}_{Aug})\}$ since the maximum rating in our experimental dataset is 5. For implicit feedback, LLM predicts disliked items, and we use them as preferred items.

ChatGPT: We predicted explicit feedback using prompts described in [10] (Details are seen in the appendix) and `gpt-3.5-turbo-0125`[6]. We did not perform predictions for implicit feedback due to costs[7].

We set parameters in RevAug as follows: $k = 100$, $d = 100$, $N = 5$, $\tau = 0.1$, $\alpha = 10^{-3}$, $\lambda = 10^{-3}$, $p = 0.5$, and $m = 5$. We used `SimCSE-RoBERTa-large`[5][8] to represent auxiliary information. RevAug was implemented using PyTorch[13] with optimization by RAdam[11]. These parameters were also applied to Heater and CLCRec as a base model of RevAug. For hyper-parameter tuning, we divided $\mathcal{R}$ with ratios of $|\mathcal{R}_{tr}| : |\mathcal{R}_{val}| = 0.8 : 0.2$ and searched β from $\{10^{-k} \mid k = 0, 1, 2, 3, 4\}$. In data augmentation with an LLM, we only warm-start users with more than or equal to three interactions to keep the quality of few-shot learning.

Table 3: MAE (left) and improvement rate compared with Baseline (right, in percentage), where bold values represents the best method in each column.

	Movie	Music	Book	Grocery
Baseline	0.857	0.735	0.759	0.670
ChatGPT	0.800 (6.7)	0.719 (2.2)	0.718 (5.3)	0.705 (-5.3)
Heater	0.806 (5.9)	0.711 (3.2)	0.775 (-2.2)	0.655 (2.2)
Ablation	0.820 (4.3)	**0.703** (4.4)	0.723 (4.7)	0.628 (6.1)
RevAug	**0.785** (8.3)	0.714 (2.9)	**0.708** (6.6)	**0.626** (6.5)

4.2 Performance Evaluation (RQ1)

4.2.1 Explicit Feedback. We evaluated the performance of each method using MAE and its improvement rate over the Baseline (average rating)[9]. As shown in Table 3, RevAug outperformed Heater, ChatGPT, and Ablation in all datasets except the Music dataset. Focusing on the Music dataset, RevAug did not outperform them, but the difference was minor. These results suggest RevAug solved cold-start recommendations with explicit feedback by generating data through reverse prompts.

4.2.2 Implicit Feedback. We evaluated the performance of each method using nDCG@10 and its improvement rate over the Baseline (random sorting). As shown in Table 4, RevAug achieved the best results for all datasets. In the Movie and Grocery dataset, CLCRec failed training because its improvement rates were negative; on the other hand, RevAug significantly improved its performance, which

[4]MAE is defined by score($\mathcal{R}$) $:= \mathbb{E}_{(u,i)\in\mathcal{R}}[|r_{u,i} - \mathbf{f}_u^T \mathbf{f}_i|]$. nDCG@$K$ score is calculated from items whose preference scores $\hat{p}(\mathbf{f}_u, \mathbf{f}_i)$ for all items I rank in the top-K.

[5]https://nijianmo.github.io/amazon/index.html

[6]https://platform.openai.com/docs/models

[7]As will be discussed in Section 4.4, processing times for implicit feedback prediction were estimated to be 1000 times larger than that for explicit feedback predictions.

[8]https://huggingface.co/princeton-nlp/sup-simcse-roberta-large

[9]The reported scores represent the average values obtained from three separate trials.

Table 4: nDCG@10 (left) and improvement rate compared with Baseline (right, in percentage).

	Movie	Music	Book	Grocery
Baseline	0.067	0.067	0.059	0.058
CLCRec	0.057 (-15.3)	0.085 (26.0)	0.062 (5.4)	0.019 (-67.0)
Ablation	0.093 (39.5)	0.080 (18.8)	0.078 (33.2)	0.054 (-7.4)
RevAug	**0.123** (83.8)	**0.087** (29.8)	**0.080** (36.0)	**0.085** (46.9)

suggests that it can compensate for the lack of training data with generated data. Ablation's negative improvement rates in the Grocery dataset also confirmed the importance of data augmentation quality. We confirmed RevAug has a consistent positive effect on solving the cold-start problem for implicit feedback.

4.3 Dataset Size Analysis (RQ2)

In this experiment, we investigated how the improvement rates from the Baseline changed when the number of warm-start items was increased incrementally from 100 to 500 in steps of 100. Figure 2 shows the results of the improvement rates for each dataset. Detailed numerical values are provided in the appendix.

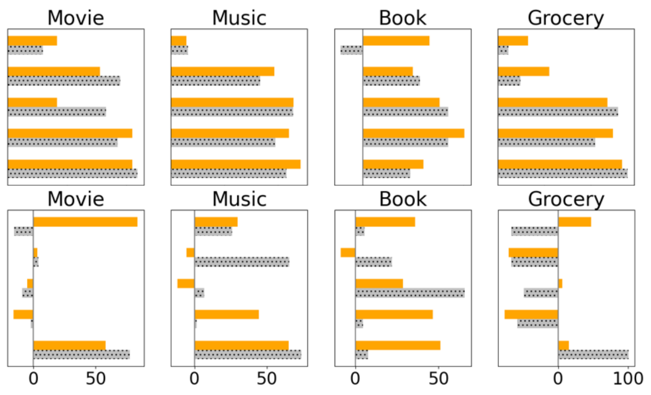


Figure 2: Improvement rates in percentage from the Baseline for RevAug (top bar, yellow) and the Base model (bottom bar, dotted gray) in cases of explicit (top figure) and implicit (bottom figure) feedback evaluations.

Table 5 shows that RevAug outperformed better than the Base model in the explicit feedback setting. While RevAug was outperformed by the Base model at 300 in the explicit feedback setting, the differences in improvement rates was slight. In the implicit feedback setting, while the improvement rates varied on datasets (Figure 2), RevAug outperformed the Base model on average except for 200 and 500 training size. Overall, we confirmed that pseudo-samples generated by RevAug enhanced the Base model's performance through data augmentation when there was insufficient warm-start data.

4.4 Cost Analysis (RQ3)

Finally, we examined the operational costs associated with an LLM. As discussed in the introduction, we focus only on inference costs. Since user features $\mathbf{f}_u$ are computed during the training phase, the inference costs for RevAug only involve calculating cold-start item

Table 5: Each entry $a/b(c)$ represents (a) the number of times RevAug exceeded the Baseline, (b) the number of times RevAug exceeded the Base model, and (c) the average of differences of improvement rates over the Baseline from that of the Base model for each dataset in percentage.

	100	200	300	400	500
Explicit	**4/3 (3.8)**	**4/2 (1.2)**	4/1 (-2.8)	**4/4 (2.6)**	**4/2 (0.5)**
Implicit	**4/4 (61.8)**	1/0 (-26.4)	2/2 **(0.8)**	2/2 **(13.3)**	**4/1** (-17.4)

Table 6: Processing time (in seconds).

		Movie	Music	Book	Grocery
RevAug	(Augmentation)	1596	413	337	111
	Inference	**0.302**	**0.077**	**0.071**	**0.041**
ChatGPT	Inference	1570	366	377	191

features $\mathbf{f}_i$ and predicting rating or preference scores via matrix operations. Although ChatGPT needs no learning costs, predicting all user-item pairs during the inference phase requires high costs. We computed processing time statistics for explicit feedback during the inference phase for RevAug and ChatGPT and during the data augmentation phase for RevAug.

Table 6 shows that the inference times of RevAug were over 1000 times faster than that of ChatGPT[10]. During the learning phase, RevAug uses LLM only for data augmentation, and the cost of LLM related to RevAug remains constant regardless of the number of inference targets. In contrast, using ChatGPT increases costs as the number of inference targets increases. It is especially critical with implicit feedback cases, which require predicting all user-item interactions. For example, the Movie dataset required 3069 inferences for explicit feedback from Table 2; however, inference for implicit feedback would be needed for all 3000×100 user-item pairs. Conducting such extensive inferences is not practical in terms of real-time analysis.

In the case of explicit feedback prediction on the Movie dataset, its API fees by `gpt-3.5-turbo-0125` took 0.19 USD for training RevAug and 0.24 USD for inference by ChatGPT. In the case of implicit feedback prediction, ChatGPT will take over 24 USD and 1.5×10^6 seconds (about 2.7 hours) only for a single small dataset. If we assume the number of cold-start items was ten times increased, ChatGPT will also be ten times the inference time and cost (240 USD and 27 hours). In contrast, RevAug will achieve rapid inference within approximately 600 seconds without additional API fees of more than 0.19 USD during inference. In summary, RevAug takes advantage of performing highly accurate inference at minimal cost.

5 CONCLUSION AND DISCUSSION

In this paper, we proposed a novel method, RevAug, for enhancing cost-efficient cold-start recommendation systems with limited warm-start training data. RevAug generates pseudo-samples using LLMs and trains a recommendation system with them. Our numerical experiments confirmed that RevAug archived higher cold-start

[10]We measured the processing time of inferences by RevAug on a CPU (not strong GPU): Ubuntu 16.04.7 LTS (GNU/Linux 4.4.0-203-generic x86_64) Intel(R) Xeon(R) CPU E7-8890 v4 @ 2.20GHz.

recommendation performances with fewer warm-start training data. Additionally, we verified that significant cost reductions are possible compared to LLM-based predictions.

Acknowledgements

We are especially grateful to Takehiro Kozushi for motivating this work in industrial contexts and the anonymous reviewers for useful comments and discussions.

REFERENCES

[1] Haoyue Bai, Min Hou, Le Wu, Yonghui Yang, Kun Zhang, Richang Hong, and Meng Wang. 2023. GoRec: A Generative Cold-start Recommendation Framework. In *Multimedia*. ACM, 1004–1012.
[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 1597–1607.
[3] Yu Cui, Feng Liu, Pengbo Wang, Bohao Wang, Heng Tang, Yi Wan, Jun Wang, and Jiawei Chen. 2024. Distillation Matters: Empowering Sequential Recommenders to Match the Performance of Large Language Model. *CoRR* abs/2405.00338 (2024).
[4] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *RecSys*. ACM, 101–109.
[5] Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple Contrastive Learning of Sentence Embeddings. In *EMNLP (1)*. ACM, 6894–6910.
[6] Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023. Chat-REC: Towards Interactive and Explainable LLMs-Augmented Recommender System. *CoRR* abs/2303.14524 (2023).
[7] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised Contrastive Learning. In *NeurIPS*.
[8] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.
[9] Dairui Liu, Boming Yang, Honghui Du, Derek Greene, Aonghus Lawlor, Ruihai Dong, and Irene Li. 2023. RecPrompt: A Prompt Tuning Framework for News Recommendation Using Large Language Models. *CoRR* abs/2312.10463 (2023).
[10] Junling Liu, Chao Liu, Renjie Lv, Kang Zhou, and Yan Zhang. 2023. Is ChatGPT a Good Recommender? A Preliminary Study. *CIKM 2023 GenRec Workshop* abs/2304.10149 (2023).
[11] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020. On the Variance of the Adaptive Learning Rate and Beyond. In *ICLR*. OpenReview.net. https://openreview.net/forum?id=rkgz2aEKDr
[12] Jianmo Ni, Jiacheng Li, and Julian J. McAuley. 2019. Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. In *EMNLP/IJCNLP (1)*. ACM, 188–197.
[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*. 8024–8035.
[14] Steffen Rendle, Walid Krichene, Li Zhang, and John R. Anderson. 2020. Neural Collaborative Filtering vs. Matrix Factorization Revisited. In *RecSys*. ACM, 240–248.
[15] Scott Sanner, Krisztian Balog, Filip Radlinski, Ben Wedin, and Lucas Dixon. 2023. Large Language Models are Competitive Near Cold-start Recommenders for Language- and Item-based Preferences. In *RecSys*. ACM, 890–896.
[16] Zhongxiang Sun, Zihua Si, Xiaoxue Zang, Kai Zheng, Yang Song, Xiao Zhang, and Jun Xu. 2024. Large Language Models Enhanced Collaborative Filtering. *CoRR* abs/2403.17688 (2024).
[17] Maksims Volkovs, Guang Wei Yu, and Tomi Poutanen. 2017. DropoutNet: Addressing Cold Start in Recommender Systems. In *NeurIPS*. 4957–4966.
[18] Jianling Wang, Haokai Lu, James Caverlee, Ed H. Chi, and Minmin Chen. 2024. Large Language Models as Data Augmenters for Cold-Start Item Recommendation. In *WWW*. ACM, 726–729.
[19] Yinwei Wei, Xiang Wang, Qi Li, Liqiang Nie, Yan Li, Xuanping Li, and Tat-Seng Chua. 2021. Contrastive Learning for Cold-Start Recommendation. In *ACM Multimedia*. ACM, 5382–5390.
[20] Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, Hui Xiong, and Enhong Chen. 2023. A Survey on Large Language Models for Recommendation. *CoRR* abs/2305.19860 (2023).
[21] Xuansheng Wu, Huachi Zhou, Wenlin Yao, Xiao Huang, and Ninghao Liu. 2023. Towards Personalized Cold-Start Recommendation with Prompts. *CoRR* abs/2306.17256 (2023).
[22] Yuhui Zhang, Hao Ding, Zeren Shui, Yifei Ma, James Zou, Anoop Deoras, and Hao Wang. 2021. Language Models as Recommender Systems: Evaluations and Limitations. In *I (Still) Can't Believe It's Not Better! NeurIPS 2021 Workshop*. OpenReview.net. https://openreview.net/forum?id=hFx3fY7-m9b
[23] Zhihui Zhou, Lilin Zhang, and Ning Yang. 2023. Contrastive Collaborative Filtering for Cold-Start Item Recommendation. In *WWW*. ACM, 928–937.
[24] Yaochen Zhu, Liang Wu, Qi Guo, Liangjie Hong, and Jundong Li. 2024. Collaborative Large Language Model for Recommender Systems. In *WWW*. ACM, 3162–3172.
[25] Ziwei Zhu, Shahin Sefati, Parsa Saadatpanah, and James Caverlee. 2020. Recommendation for New Users and New Items via Randomized Training and Mixture-of-Experts Transformation. In *SIGIR*. ACM, 1121–1130.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009