Enhancing conversational recommendation systems through the integration of KNN with ConLinUCB contextual bandits

Yisen Zhao

School of Information Science and Engineering, East China University of Science and Technology, Shanghai, China

21011524@mail.ecust.edu.cn

Abstract. In recommender system research, contextual multi-armed bandits have shown promise in delivering tailored recommendations by utilizing contextual data. However, their effectiveness is often curtailed by the cold start problem, arising from the lack of initial user data. This necessitates extensive exploration to ascertain user preferences, consequently impeding the speed of learning. The advent of conversational recommendation systems offers a solution. Through these systems, the conversational contextual bandit algorithm swiftly learns user preferences for specific key-terms via interactive dialogues, thereby enhancing the learning pace. Despite these advancements, there are limitations in current methodologies. A primary issue is the suboptimal integration of data from key-term-centric dialogues and arm-level recommendations, which could otherwise expedite the learning process. Another crucial aspect is the strategic suggestion of exploratory key phrases. These phrases are essential in quickly uncovering users' potential interests in various domains, thus accelerating the convergence of accurate user preference models. Addressing these challenges, the ConLinUCB framework emerges as a groundbreaking solution. It ingeniously combines feedback from both arm-level and key-term-level interactions, significantly optimizing the learning trajectory. Building upon this, the framework integrates a K-nearest neighbour (KNN) approach to refine key-term selection and arm recommendations. This integration hinges on the similarity of user preferences, further hastening the convergence of the parameter vectors.

Keywords: Conversational contextual bandits, ConLinUCB, KNN.

1. Introduction

In the expansive field of personalized recommendation systems, a key research priority has been to tackle the cold start problem and enhance the quality of recommendations. Traditional systems, reliant on analyzing historical user data to predict future interests, face challenges when dealing with new users or items where such data is minimal or absent. Addressing these challenges, bandit algorithms have become integral, adeptly balancing exploration—suggesting new items to gather fresh feedback and gain insight into user preferences—and exploitation—recommending items based on existing interactions. The crucial aspect lies in striking a balance: too much exploration can lead to suboptimal results, while excessive exploitation might miss potentially interesting items. Contextual bandit approaches, which incorporate feature-based learning, offer significant improvements in recommendation capabilities, particularly for new users or items. However, these methods often overlook the potential benefits of collaborative user feedback. To harness this untapped resource, Gentile et al.'s CLUB algorithm utilizes user clustering, Li et al. extended this with COFIBA's collaborative clustering, and Gentile et al.'s CAB algorithm further refines it by combining user and content clustering [1,2].

Recently, conversational recommendation systems have gained prominence. These systems actively engage users in dialogues, gathering dynamic feedback on preferences. This interactive method is especially effective against the limitations of traditional systems in handling the cold-start problem. Despite their potential, contextual bandit algorithms require extensive exploration to collect feedback for optimal results. To streamline this, researchers introduced the conversational contextual bandit algorithm, incorporating the concept of key-terms. This allows for dynamic understanding of user preferences through dialogue feedback. However, this approach has yet to fully utilize the interplay between key-term dialogue and arm-level recommendations, leaving room for improved learning efficiency.

In response to this, our paper introduces the ConLinUCB-KNN method, an innovation that builds upon the ConLinUCB framework. This approach combines the strengths of ConLinUCB in integrating specific interactions and broader queries with a KNN strategy that harnesses the relevance of key-terms. This synergistic combination not only promises to mitigate the cold-start problem by rapidly developing an accurate model of user preferences but also introduces a level of adaptability and precision previously unattainable in recommendation systems [3].

2. Method

2.1. *ConLinUCB in Conversational Recommender Systems*

This section begins by discussing the problem setting for conversational contextual bandits. Assume there exists a finite set A of actions, with each arm $a \in A$ representing an item to be recommended. Each arm a is connected with a normalized contextual vector $\mathbf{x}_t^a \in \mathbb{R}^d$, where the normalization constrains the vector such that its Euclidean norm $\| \mathbf{x}_t^a \|_2$ dose not exceed 1, ensuring a standardized scale for comparison [4]. The user's preferences for items are captured by an unknown vector θ^*, $\| \theta^* \|_2 \leq 1$. The interaction between the agent and a user unfolds over a finite series of rounds $T \in \mathbb{N}$, with each round $t = 1,2,\dots,T$ offering a subset of arms $A_t \subseteq A$ for selection. Choices are informed by historical interactions, culminating in the selection of an arm a_t from A_t and the acquisition of a corresponding reward $r_{a,t}$, which is postulated as a linear function of the contextual vectors

$$r_{a,t} = \mathbf{x}_t^{a^\top} \theta^* + \epsilon_t \tag{1}$$

And ϵ_t denotes sub-Gaussian random noise with zero mean. The primary learning objective is to minimize cumulative regret $R(T)$, a metric quantifying the lost reward opportunity by not always choosing the optimal action, defined as the difference between the sum of rewards from the chosen actions and the optimal actions over time:

$$R(T) = \sum_{t=1}^{T} \max_{a \in A_t} \mathbf{x}_t^{a^\top} \theta^* - \sum_{t=1}^{T} \mathbf{x}_t^{a_t^\top} \theta^* \tag{2}$$

The agent can intermittently ask certain key-terms during conversations to elicit user preferences more effectively. A "key term" refers to a keyword or topic that is connected to a subset of arms. Let's consider there's a defined set K of such key-terms [5]. The interrelationship between the arms and key-terms is characterised by a weighted bipartite graph (A, K, W), with $W = \{w_{a,k} | a \in A, k \in K\}$ denoting the weight representing the connection between an arm a and a key-term k, where each $w_{a,k} \geq 0$. It is assumed that each key-term k is linked to at least one arm a with a positive weight, i.e., $\sum_{a \in A} w_{a,k} > 0$

for all $k \in K$, and the sum of weights for each arm totals to one, that is $\sum_{k \in K} w_{a,k} = 1$ for any $a \in A$. The feature vector for a key-term k is formulated as:

$$\tilde{x}_k = \sum_{a \in A} \sum_{a' \in A} w_{a',k} \cdot x_{a'} \tag{3}$$

Feedback at the level of key-terms $r_{k,t}$, is defined by $r_{k,t} = \tilde{x}_k^{\top} \theta^* + \tilde{\epsilon}_t$, where $\tilde{\epsilon}_t$ is presumed to be sub-Gaussian noise with a zero mean. To preserve user experience, the agent should not engage in conversations too frequently. And a function $b: \mathbb{N}_+ \to \mathbb{R}_+$ is defined to manage the frequency of the agent's conversations, with $b(t)$ progressively increasing with t to moderate the conversation rate. At each round t, if $b(t) - b(t-1) > 0$, the agent is allowed to conduct $q(t) = b(t) - b(t-1)$ conversations by soliciting user feedback on $q(t)$ key-terms. The agent will engage in $b(t)$ conversations with the user up to round t by using this paradigm modeling arrangement [6].

In conversational bandit, existing research cannot fully exploit information about both arm-level and key-term-level. Therefore, this paper introduces ConLinUCB, a conversational framework designed to enhance information integration and is compatible with a wide range of key-term selection strategies.

The ConLinUCB framework orchestrates a conversational recommendation system by adaptively refining its recommendations through a systematic learning of user preferences. Firstly, the graph (A, K, W), conversation frequency function $b(t)$ and key-term selection strategy are input [7]. Then, the algorithm initially establishes key matrices and vectors, which are updated in successive rounds of interaction. Utilizing the function $b(t)$, the algorithm moderates its queries for user feedback, engaging the user with strategically chosen key-terms when the situation warrants. The user's feedback informs updates to the matrix M_t and vector b_t, aggregating valuable insights into the user's preferences over time. Leveraging this data, the algorithm employs an upper confidence bound (UCB) strategy to select the most fitting arm a_t in each round, guided by the goal of aligning with the user's evolving preferences. Subsequent to the arm choice, user feedback is recorded as a reward $r_{a,t}$, further honing the algorithm's predictive accuracy. By iterating this process, the ConLinUCB framework aspires to incrementally enhance personalization and user satisfaction, synthesizing user preference comprehension with calculated exploration.

2.2. KNN for Enhanced Similarity-Based Recommendations

In the context of conversational bandit frameworks for recommendation systems, the key-term selection strategy plays a pivotal role by aligning user interactions with the system's recommendations, thereby enhancing the personalization of the content suggested [8]. Employing the K-Nearest Neighbors (KNN) strategy within the ConLinUCB framework serves as an innovative mechanism for key-term selection, allowing for a nuanced approach that leverages user preference data more effectively. The core idea of this strategy is to leverage the relevance between key-terms associated with content items and user preferences to accurately identify and recommend content that matches user interests [9].

The ConLinUCB-KNN algorithm is shown in Table 1.

Step 1: The ConLinUCB-KNN algorithm begins by initializing a covariance matrix M based on a parameter β, along with a reward vector b initially set to zero.

Step 2: For each time step t, the algorithm checks if the current session frequency function $b(t)$ is greater than the previous time point, determining whether to enter the key-term exploration phase. If $b(t) - b(t-1) > 0$ then let $q(t) = b(t) - b(t-1)$.

If $q(t) > 0$, for each content item in the recommendation system, its relevance is assessed by computing the similarity $s(k)$ between it and the user's current preference vector θ using the dot product formula

$$s(k) = \theta^T x_k \tag{4}$$

Where x_k represents the feature vector of the key-term.

Subsequently, based on the computed similarity scores, the top k items are selected with the highest similarity from all content items, forming a candidate set. After trying different k values through multiple experiments, when k=5, the computational burden can be reduced better and better performance can be provided. These selected items represent the options that are most closely aligned with the user's current preferences [10].

Further, to comprehensively consider the information of these candidate items, the weighted average feature vector of these items' feature vectors are calculated, as shown in the following formula

$$\bar{x} = \frac{\sum_{i=1}^{k} s_i \cdot x_{\mathrm{k}_i}}{\sum_{i=1}^{k} s_i} \tag{5}$$

Where s_i and x_{k_i} respectively denote the similarity score and the corresponding feature vector of the ith candidate item.

The expected total reward (pta) for each candidate item uses the dot product of the weighted average feature vector $\bar{x}$ and the user's preference vector θ, and select the item with the highest pta value as the final recommendation:

$$pta = \max(\theta^T \bar{x}) \tag{6}$$

The user's preference for the set of key-terms represented by this weighted average feature vector is queried to collect feedback [11].

The user's feedback $r_{k,t}$ is used to update the covariance matrix M and the reward vector b, reflecting new information about user preferences.

If $q(t) = 0$, then there is no conversation recommendation for key-terms in this round, and the algorithm proceeds directly to the update phase.

Step 3: At the end of each time step, the algorithm calculates the parameter θ through the product of the inverse of M and the reward vector b, then selects the arm a_t that maximizes the dot product of θ with the candidate arms plus a confidence upper bound. The user's preference feedback for the selected arm a_t is used for the final updates of M and b, enabling the ConLinUCB-KNN algorithm to continually learn user preferences through real-time feedback and optimize key-term selection using the KNN strategy, thereby enhancing the relevance and precision of personalized recommendations.

To be specific, within the ConLinUCB approach at each round t, the algorithm utilizes a process designed to prevent overfitting through regularization, aligning the system's predictions with the feedback received from users by estimating the user preference vector via a linear regression process.

$$\theta_t = \underset{\theta \in \mathbb{R}^d}{\operatorname{argmin}} \sum_{\tau=1}^{t-1} (\mathbf{x}_{a_\tau}^\top \theta - r_{a_\tau,\tau})^2 + \sum_{\tau=1}^{t} \sum_{k \in K_\tau} (\tilde{\mathbf{x}}_k^\top \theta - \tilde{r}_{k,\tau})^2 + \beta \parallel \theta \parallel_2^2 \tag{7}$$

Where K_τ represents the set of key-terms asked at round τ, and the coefficient $\beta > 0$ controls regularization. The closed-form solution to this optimization issue is $(\theta_t = M_t^{-1} b_t)$, where.

$$\begin{gathered} M_t = \sum_{\tau=1}^{t-1} \mathbf{x}_{a_\tau} \mathbf{x}_{a_\tau}^\top + \sum_{\tau=1}^{t} \sum_{k \in K_\tau} \tilde{\mathbf{x}}_k \tilde{\mathbf{x}}_k^\top + \beta I \\ b_t = \sum_{\tau=1}^{t-1} \mathbf{x}_{a_\tau} r_{a_\tau,\tau} + \sum_{\tau=1}^{t} \sum_{k \in K_\tau} \tilde{\mathbf{x}}_k \tilde{r}_{k,\tau} \end{gathered} \tag{8}$$

ConLinUCB uses an upper bound confidence (UCB) strategy to select arms to balance exploration and exploitation.

$$a_t = \underset{a \in A_t}{\operatorname{argmax}} \mathbf{x}_a^\top \theta_t + \alpha_t \parallel \mathbf{x}_a \parallel_{M_t^{-1}} = \hat{R}_{a,t} + C_{a,t} \tag{9}$$

Where $\| \mathbf{x}_a \|_{M_t^{-1}} = \sqrt{\mathbf{x}_a^\top M_t^{-1} \mathbf{x}_a}$, $\hat{R}_{a,t}$ and $C_{a,t}$ represent the estimated reward and confidence radius of arm a at round t, and.

$$\alpha_t = \sqrt{2\log\left(\frac{1}{\delta}\right) + d\log\left(1 + \frac{t+b(t)}{\beta d}\right)} + \sqrt{\beta} \tag{10}$$

Table 1. ConLinUCB-KNN

```
Algorithm 1: ConLinUCB-KNN
Input: graph$(A, K, W)$, conversation frequency function $b(t)$.
Initialization: $M_0 = \beta I, b_0 = 0$.
for $t = 1 \rightarrow T$ do
if $b(t) - b(t-1) > 0$ then
$q(t) = b(t) - b(t-1)$;
While $q(t) > 0$ do
For each key-term $k$, its similarity: $s(k) = \theta^T x_k$;
Select top $k$ key-terms with the highest similarity for querying to calculate the weighted average feature vector $\bar{x}$ of these items' feature vectors: $\bar{x} = \frac{\sum_{i=1}^{k} s_i \cdot x_{k_i}}{\sum_{i=1}^{k} s_i}$
expected total reward: $pta = \max(\theta^T \bar{x})$, and ask the user's preference;
Receive the user's feedback $\tilde{r}_{k,t}$;
$M_t = M_{t-1} + \tilde{x}_k \tilde{x}_k^\top$;
$b_t = b_{t-1} + \tilde{x}_k^\top r_{k,t}$;
$q(t) = q(t) - 1$;
end while
else
$M_t = M_{t-1}, b_t = b_{t-1}$;
end if
$\theta_t = M_t^{-1} b_t$;
Select $a_t = \underset{a \in A_t}{\operatorname{argmax}} \mathbf{x}_a^\top \theta_t + \alpha_t \| \mathbf{x}_a \|_{M_t^{-1}}$;
Ask the user's preference on arm $a_t$ and receive the reward $r_{a,t}$;
$M_t = M_{t-1} + x_{a_t} x_{a_t}{}^T$;
$b_t = b_{t-1} + x_{a_t} r_{a_t,t}$;
end for
```

3. Experiments on Synthetic Dataset

This section verifies the effectiveness of the ConLinUCB algorithm on synthetic datasets.

Generation of the synthetic dataset. Initially, a synthetic dataset comprising is generated a key-term set $\mathcal{K}$ with $|\mathcal{K}| = 500$ and an arm set $\mathcal{A}$ with $|\mathcal{A}| = 5{,}000$ arms. Each arm a in $\mathcal{A}$ is connected with a d-dimensional feature vector x_a and a corresponding subset of key-terms V_a from $\mathcal{K}$; each key-term in the subset has an equal effect weight of $\frac{1}{|V_a|}$. The construction of the arm feature vectors proceeds through the following steps:

For each k in $\mathcal{K}$, a pseudo-feature vector $\tilde{x}_k$ is generated, with each dimension drawn from a uniform distribution $U(-1,1)$.

For each arm a, a quantity n_a of key-terms from $\mathcal{K}$ is uniformly and randomly selected, assigning them into the set V_a. Each key-term is allotted an equal weight of $\frac{1}{n_a}$.

The final feature vector x_a for each arm is computed by aggregating the weighted pseudo-feature vectors of its key-terms, normalized by n_a and a noise factor σ_g that follows an independent Gaussian distribution for each dimension.

Subsequently, N_u users are generated, with each user u having a preference vector θ_u, where every dimension is uniformly distributed according to $U(-1,1)$. It is important to note that while the user's preference vector is crafted to encapsulate the actual preferences, it also aligns with the constructed feature vectors of the arms [12].

The system provides a limited selection of arms from $\mathcal{A}$, denoted as $\mathcal{A}_t$, from which users randomly choose. Feedback is then gathered based on the user's selections and corresponding key-term reactions, following predefined equations. This feedback, influenced by a Gaussian noise component with a standard deviation of σ_g, serves as the basis for refining the recommendation model. For this scenario, the parameters are set to: $N_u = 200, N = 5000, K = 500$, subset arm $\mathcal{A}_t$ size of 50, $\sigma_g = 0.1$.

Baselines. The ConLinUCB-KNN algorithm presented in this paper will be compared with the following baselines:

LinUCB: A highly regarded contextual multi-armed bandit algorithm that posits a linear relationship between the latent features of users and items and the obtained reward. It solely relies on feedback from the selected arm, omitting conversational data [13].

Arm-Con: An extension of the LinUCB algorithm, this method instigates a dialogue by inquiring whether the user favours the arm selected by the bandit. Arm-Con incorporates feedback from these conversations while continuing to utilize LinUCB for the selection of arms [14].

ConUCB: This introduces the inaugural conversational algorithm designed for the multi-armed bandit framework. It takes into account not just the feedback from arms but also garners insights from conversational interactions regarding key-terms [15].

Evaluation Results. The experiment assumes total rounds $T = 1{,}000$, the frequency function $b(t) = 5\lfloor log(t+1) \rfloor$ and the number of arms $|\mathcal{A}_t| = 50$ and evaluates the changes in regret and user preference feature vectors θ of the four algorithms as the number of rounds increases.

Run the experiments ten times and calculate the average regret of all users for these four algorithms. The results are shown in Figure 1. First, all other algorithms exceed LinUCB, demonstrating the value of communication. The regret based on the ConLinUCB framework combined with the KNN method is the lowest among the four algorithms, far lower than the other algorithms.

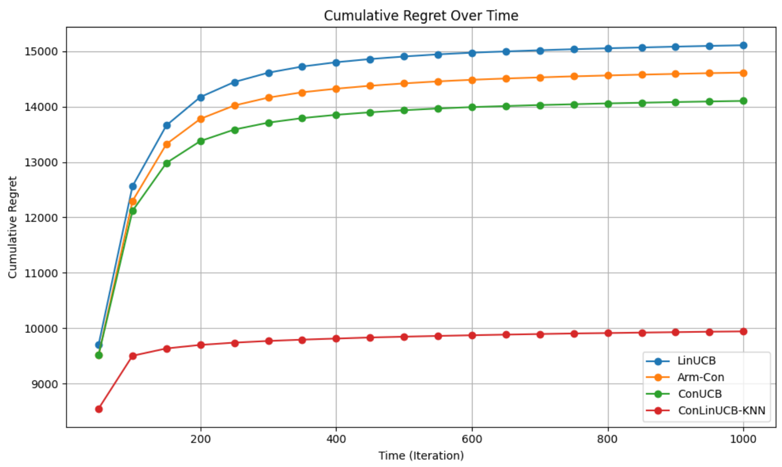

Figure 1. Cumulative Regret on Synthetic dataset (Photo/Picture credit: Original).

At the same time, the experiment compares the changes in user preference feature vectors θ in the four algorithms, as shown in Figure 2. It is not difficult to see that the changes in theta of the three algorithms LinUCB, Arm-Con, and ConUCB are basically the same. On the contrary, in ConLinUCB-KNN, the convergence speed of theta is significantly faster.

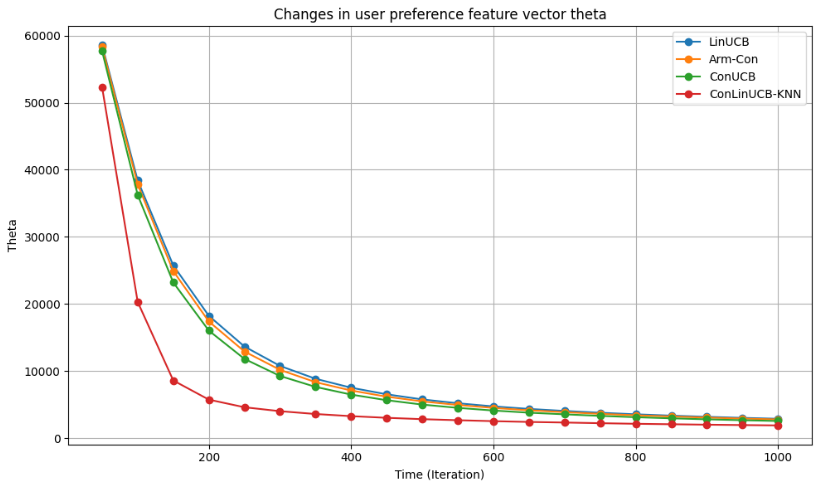

Figure 2. User Preference Feature Vector on Synthetic dataset (Photo/Picture credit: Original).

Based on the above analysis, the following conclusions can be given:

Compared to the accumulated regrets among the four algorithms, it can be observed that Arm-Con, ConUCB, and ConLinUCB-KNN outperform LinUCB due to their incorporation of conversational strategies. Similarly, since ConLinUCB-KNN combines the ConLinUCB framework with the KNN strategy, which offers a more flexible key-term selection policy, its performance surpasses that of ConUCB and Arm-Con.

The changes in user preference feature vectors indicate that ConLinUCB-KNN exhibits faster learning speed and better convergence performance compared to the other three algorithms. It demonstrates the ability to learn users' true preferences more rapidly and shows good stability.

4. Discussion

The comparative analysis of accumulated regrets among the tested algorithms yields valuable insights into the efficacy of conversational enhancements in recommendation systems. The integration of conversational strategies within the recommendation algorithms, as exemplified by Arm-Con, ConUCB, and particularly ConLinUCB-KNN, significantly enhances their performance over the traditional LinUCB algorithm. This enhanced performance can be largely attributed to the conversational component's ability to capture and utilize nuanced user preferences that are often not evident from implicit user interactions alone.

ConLinUCB-KNN, with its hybrid approach combining the robustness of the ConLinUCB framework and the flexibility of the KNN strategy, stands out by offering a dynamic and nuanced key-term selection strategy. Beyond improving the algorithm's accuracy, this strategy augments the user experience by offering more nuanced and personalized recommendations, thus better aligning with the user's evolving interests and needs.

Significant performance improvements, evidenced by the algorithm's faster learning rates and improved convergence, underscore its ability to rapidly model and adapt to users' true preferences—an advantage in domains with rapidly changing user interests. This rapid adaptability is essential in fast-paced domains where user preferences evolve swiftly, such as trending topics in social media or personalized shopping experiences.

Despite its successes, one cannot overlook the complexity of ConLinUCB-KNN, as its computational demands scale with the size of the key-term and action spaces, potentially posing scalability challenges in extensive applications. Another concern is the assumption of readily available high-quality feedback. In practice, user feedback can be inconsistent and prone to noise, making it less reliable for training sophisticated models. Such scenarios necessitate the development of robust mechanisms within the algorithm to mitigate the impact of unreliable feedback.

For future advancements, it is crucial to explore machine learning techniques like ensemble methods, which could integrate multiple models to refine the recommendation quality further. Additionally, investigating ways to enhance the interpretability of the algorithm could foster trust and transparency in AI-driven recommendation systems.

5. Conclusion

This paper presents the ConLinUCB-KNN algorithm, a novel fusion of the ConLinUCB framework with K-nearest neighbors methodology, marking a significant advancement in the realm of recommendation systems. Our empirical investigations, conducted on a synthetic dataset, underscore the algorithm's proficiency in contextual information-rich recommendation tasks, exemplified by its rapid adaptation to user preferences inferred from behavioral data. A key innovation lies in the use of KNN for in-depth key-term similarity analysis, significantly aligning the algorithm’s predictions with users' actual preferences, thereby refining the recommendation process. Notwithstanding its strengths, the ConLinUCB-KNN algorithm faces challenges concerning computational complexity, particularly noticeable with expanding key-term sets and an increasing number of candidate arms. This aspect presents a potential bottleneck, especially in large-scale deployments, and necessitates further optimization for practical applications. Despite these hurdles, the ConLinUCB-KNN algorithm represents a substantial leap forward in personalized recommendation systems, chiefly in elevating the precision of user-specific suggestions. Looking ahead, there is fertile ground for research in integrating additional machine learning methodologies to enhance the algorithm’s performance. Potential avenues include incorporating deep learning for more nuanced feature extraction and leveraging reinforcement learning to refine the algorithm’s decision-making process. Crucially, addressing the algorithm's scalability and reducing computational demands for large datasets stand out as imperative directions for future research, aiming to bolster both its practicality and efficiency.

References

[1] Lu, J., Wu, D., Mao, M., Wang, W., & Zhang, G. (2015). Recommender system application developments: A survey. Decision Support Systems, 74, 12–32. https://doi.org/10.1016/j.dss.2015. 03.008.

[2] Elahi, M., Ricci, F., & Rubens, N. (2016). A survey of active learning in collaborative filtering recommender systems. Computer Science Review, 20, 29–50. https://doi.org/10.1016/j.cosrev.2016. 05.002.

[3] Chen, Y., Vankov, E., Baltrunas, L., Donovan, P., Mehta, A., Schroeder, B., & Herman, M. (2023). Contextual multi-armed bandit for email layout recommendation. In Proceedings of the 17th ACM Conference on Recommender Systems (RecSys '23) (pp. 400–402). Association for Computing Machinery. https://doi.org/10.1145/3604915.3608878.

[4] Walsh, T. J., Szita, I., Diuk, C., & Littman, M. L. (2009). Exploring compact reinforcement-learning representations with linear regression. In Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI '09) (pp. 591–598). AUAI Press.

[5] Gentile, C., Li, S., & Zappella, G. (2014). Online clustering of bandits. In Proceedings of the 31st International Conference on Machine Learning (ICML'14) (Vol. 32, pp. II–757–II–765). JMLR.org.

[6] Li, S., Karatzoglou, A., & Gentile, C. (2016). Collaborative Filtering Bandits. https://doi.org/10. 1145/ 2911451. 2911548.

[7] Gentile, C., Li, S., Kar, P., Karatzoglou, A., Zappella, G., & Etrue, E. (2017). On context-dependent clustering of bandits. In Proceedings of the 34th International Conference on Machine Learning (ICML'17) (Vol. 70, pp. 1253–1262). JMLR.org.

[8] Zhu, X., Huang, Y., Wang, X., & Wang, R. (2023). Emotion recognition based on brain-like multimodal hierarchical perception. Multimedia Tools and Applications, 1-19.

[9] Jannach, D., Manzoor, A., Cai, W., & Chen, L. (2021). A survey on conversational Recommender Systems. ACM Computing Surveys, 54(5), 1–36. https://doi.org/10.1145/3453154

[10] Gao, C., Lei, W., He, X., De Rijke, M., & Chua, T. (2021). Advances and challenges in conversational recommender systems: A survey. AI Open, 2, 100–126. https://doi.org/10.1016/j.aiopen.2021.06. 002.
[11] Zhang, X., Xie, H., Li, H., & Lui, J. C. S. (2020). Conversational Contextual Bandit: Algorithm and Application. https://doi.org/10.1145/3366423.3380148.
[12] Zhao, C., Yu, T., Xie, Z., & Li, S. (2022). Knowledge-aware Conversational Preference Elicitation with Bandit Feedback. Proceedings of the ACM Web Conference 2022. https://doi.org/10.1145/ 3485447.3512152.
[13] Wang, Z., Liu, X., Li, S., & Lui, J. C. S. (2023). Efficient explorative key-term selection strategies for conversational contextual bandits. In Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence (AAAI'23/IAAI'23/EAAI'23) (Vol. 37, Article 1156, pp. 10288–10295). AAAI Press. https://doi. org/10.1609/aaai. v37i8.26225.
[14] Xia, H., Lu, Z., & Hong, W. (2022). A Multi-Armed Bandit Recommender Algorithm Based on Conversation and KNN. https://doi.org/10.1145/3579654.3579714.
[15] Christakopoulou, K., Radlinski, F., & Hofmann, K. (2016). Towards Conversational Recommender Systems. https://doi.org/10.1145/2939672.2939746.