IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 25, 2022, accepted June 6, 2022, date of publication June 13, 2022, date of current version June 17, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3182390

An End-to-End Personalized Preference Drift Aware Sequential Recommender System With Optimal Item Utilization

SARANYA MANEEROJ AND NAKARIN SRITRAKOOL

Advanced Virtual and Intelligent Computing Center, Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University, Bangkok 10330, Thailand

Corresponding author: Nakarin Sritrakool (nakarin.s@math.sc.chula.ac.th)

ABSTRACT The user preference is dynamic and requires drift detection to capture changes for delivering relevant recommendations. A sequential recommender system with drift detection was proposed, where drift points are indicated by comparing characteristics of consecutive items. The model leverages drift points to retrieve only interactions with preferences relevant to the current user preference. Nonetheless, the number of utilized items is pre-defined and may not be optimal. It is also not a unified architecture that requires optimizing each part individually. Recently, a Content-Based Transformer has been proposed to consider only items with similar characteristics by leveraging a similarity function. Content-Based Transformer is trained in an end-to-end approach and can be applied for the sequential recommendation task, where the drift of the user preference is indicated as the point where the item's group changes. However, Content-Based Transformer provides the item's group as the hard label, ignoring the item characteristics in a real-world scenario where items can exist in many groups. For instance, most movies have multiple genres. This work proposes a unified sequential recommender system that detects the personalized drift pattern of user preference. It groups similar items with soft labels and utilizes the optimal amount of items. The model is trained in an end-to-end approach to jointly optimize for group items with similar characteristics and deliver relevant recommendations. We conducted the experiments to verify the effectiveness of the proposed method by comparing it with Content-Based Transformers and related methods. The evaluation results show that the proposed method consistently outperforms the baselines.

INDEX TERMS Neural networks, machine learning, recommender systems.

I. INTRODUCTION

In online platforms with a massive number of items, the Recommender System (RS) is essential for recommending items relevant to the target user preference. Early works adopted the Collaborative Filtering approach by considering rating information and applying the Matrix Factorization (MF) to predict unobserved ratings [1]–[3]. The latent representations of user and item are extracted to calculate the inner product between both vectors for predicting missing ratings. However, the operation in this technique is linear, which limits the model to capture complex relations between users and items [4]. Several works are proposed to perform the MF by a deep neural network with non-linear functions to capture complex relationships between users and items [4], [5]. Moreover, different neural network architectures are also applied to the recommender system field, such as Auto-Encoder [6]–[8], restricted Boltzmann machines [6], [9], and Convolution Neural Networks [6], [10]. Apart from these methods, other techniques also have been applied to deliver further accurate prediction of unobserved ratings, e.g., error-based correction [11]. Although the these techniques can model the relationships between users and items, these approaches ignore the dynamics of user preference.

There is a Sequential Recommender System (SRS), which can capture user preference dynamics by considering both rating information and the sequence of the user's behavior [6]. The Recurrent Neural Network (RNN) is exploited to capture the user preference dynamics through the hidden state and utilize it for delivering the prediction [12]–[15]. Recently, several SRSs employed the Transformer architecture [16] to consider the sequence of the user's behavior [17], [18]. Transformer-based models reported better performance and delivered more relevant recommendations

The associate editor coordinating the review of this manuscript and approving it for publication was Wenbing Zhao.

than traditional RNN methods. Additionally, several works modified the self-attention mechanism in the Transformer to better suit the task of sequential recommendation [19]–[21]. For example, DSAN [19] is proposed to ignore irrelevant past interactions by leveraging a sparse activation function [22] instead of a softmax function as in typical self-attention [16]. The ignored interactions benefit DSAN by decreasing noises as irrelevant interactions to the current user preference and result in superior performance than the previous state-of-the-art. However, the mentioned RNN-based and Transformer-based methods only concern the order of the user's behavior to model user preference dynamics. They ignore the drift of the user preference in a real-world scenario where the user preference can be drifted at arbitrary points in time. Therefore, the recommendation without awareness of preference drift may not be relevant to the current user preference.

One line of work that can be applied to model the drift of user preference is Phased-LSTM [23]. This model considers the sequential information efficiently by assuming that not every interaction is important for the model's learning and can be ignored. Phased-LSTM utilizes frequency to control updates of its hidden state when considering the sequential information. Applying Phased-LSTM for the SRS task implies that users drift their preferences at an identical fixed frequency. However, Phased-LSTM does not correspond with the highly dynamic nature of user preference, where users drift their preferences differently from each other and are not characterized by frequency.

Another line of work that aims to capture the drift of user preference is Time-LSTM [24]. This model categorizes user preference into short-term and long-term preferences. Time-LSTM determines the drift of the user preference by exploiting the time interval between the consecutive interactions in the historical sequence. It assumes that a high time interval indicates the pause in the users' activity with the system, and users may drift their preferences. The recommendation from Time-LSTM is based on the long-term preference when the users drift their preferences. In contrast, the model employs the short-term preference when no drift occurred. Although the time interval may indicate the drift of the user preference, Time-LSTM utilizes the identical magnitude of the time interval to determine the preference drift of every user. Therefore, the preference drift pattern is not personalized, resulting in incorrect detection of the user preference and irrelevant recommendations.

In our previous work, we proposed an SRS called PPD [25]. It can detect the personalized drift pattern instead of employing the frequency or the time interval as in Phased-LSTM and Time-LSTM. Moreover, PPD utilizes only relevant interactions in the historical sequence to deliver the relevant next item rather than considering the whole historical sequence, which contains noises from irrelevant past interactions. As shown in Fig. 1 (a), the architecture of PPD can be divided into three main components: item representations extraction, drift detection, and next item prediction. In the first stage, PPD adopts the next item prediction task

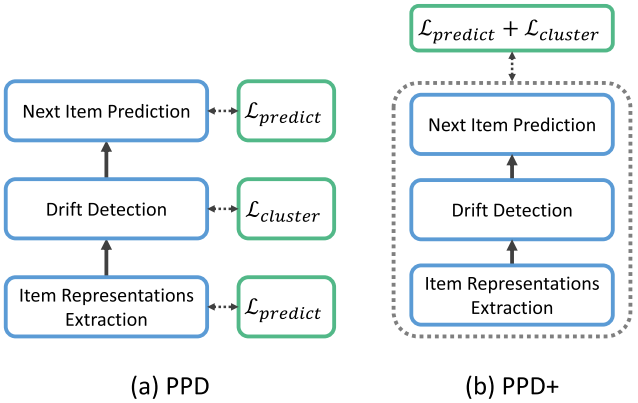


FIGURE 1. An overview of PPD and PPD+, each model consists of three main components: item representations extraction, drift detection, and next item prediction. PPD is trained with three separate components, whereas PPD+ is a unified architecture that jointly optimizes each part.

to train the model and retrieve latent item representations. In the next stage, PPD detects the personalized drift pattern by utilizing the Fuzzy c-Mean to cluster latent item representations and to output item latent characteristics. The drift points of user preference are determined as the point where the characteristics of the item representation are significantly different from its previous item. PPD divides the user's historical sequence into multiple sub-sequences according to the detected drift points. Consequently, each sub-sequence contains only a particular user preference. In the final stage, PPD delivers the recommendation relevant to the current user preference by computing the similarity between the latest sub-sequence and other sub-sequences. PPD requires a hyper-parameter called gamma to control the number of utilized items from selected sub-sequences. Only sub-sequences with significant similarity with the latest sub-sequence are considered to output the recommendation.

However, PPD cannot be trained end-to-end since each model's component needs to be optimized separately with a particular loss function, as shown in Fig. 1 (a). Consequently, PPD is not applicable in a real-world scenario due to the high computational cost when optimizing each part of the model. Furthermore, the model requires the hyper-parameter to control the number of selected sub-sequences, resulting in the number of utilized items that are not optimal and personalized. PPD must determine the number of utilized items according to the user's behavior since different users require different amounts of past information to express their preferences. For example, users with diverse preferences may require more past interactions to express their preference patterns. In contrast, users who prefer items from a few categories may need fewer past interactions to define their preference patterns.

Recently, several works inspired by the state-of-the-art performance of the Transformer and modified it to adapt for a particular task [26], [27]. One line of work is proposed to reduce the complexity of the self-attention in the Transformer by utilizing a pre-defined sparse attention

pattern. For instance, Longformer [28] establishes the attention pattern as a sliding window or a dilated sliding window to capture only relations between the target item and its neighbor items. Nonetheless, the attention pattern of Longformer is not personalized, and it selects past interactions without regarding the current user preference. Content-Based Transformer (CBT) is another line of variation that establishes the relation of items in the sequence based on their characteristics with others [29], [30]. CBT categorizes each item into a corresponding group according to their characteristics based on a hard clustering algorithm. It allows only items from the same group to establish relations with each other. When applying CBT to perform the sequential recommendation task, the change of the item's group can be viewed as a drift of the user preference. The detected drift pattern is personalized since the pattern is based on changes of the item's group in the sequence, which is unique for each user. The SRS can adopt CBT to be trained end-to-end while maintaining the mechanism to detect the personalized drift of the user preference. Nonetheless, most CBT groups similar items by utilizing a hard clustering algorithm, which allows an item to belong to one group only. Therefore, CBT ignores the characteristic of items in a real-world scenario where items can be in various groups simultaneously. For instance, a movie can have many genres.

To address the limitations of PPD, the model should be trained in an end-to-end approach as in CBT for enabling it to be applicable in a real-world scenario. The model should also overcome the limitation of hard clustering as in CBT which forces each item to belong to only one group. However, item characteristics in a real-world scenario can be categorized into various groups, which can naturally be captured by leveraging soft clustering. Additionally, the model must determine the number of utilized items based on individual users' behavior instead of requiring the hyper-parameter value. This work proposes a unified architecture of PPD, which can be optimized in an end-to-end fashion called PPD+. As shown in Fig. 1 (b), every part of PPD+ is trained jointly with a combination of loss functions from each component. PPD+ leverages the similarity of a user's behavior to establish the optimal amount of utilized items required by each user for the prediction.

The key contributions of this work are summarized as follows:

- This work proposes a unified model to capture user preference dynamics by detecting the actual drift of user preference based on changes in consecutive item characteristics. The model retrieves item characteristics based on soft clustering rather than hard clustering as in Content-Based Transformer to capture item characteristics when items belong to various groups. The model considers only groups of items with similar preferences as the current user preference for delivering relevant recommendations.
- To efficiently optimize the model in an end-to-end approach, we propose to combine loss functions of each model's component as a unified loss. The learning of the model is jointly optimized to group items with similar user preferences and deliver recommendations relevant to the current user preference.
- The proposed method determines the optimal number of utilized items required for the prediction based on each user's behavior rather than the pre-defined value as a hyper-parameter. The optimal item utilization allows the model to personalize and correspond to the dynamic of user behavior.

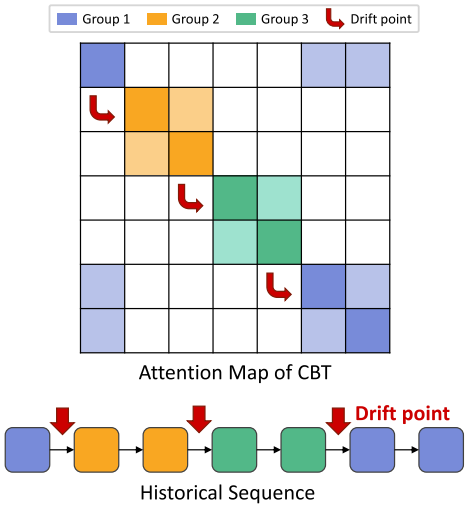


FIGURE 2. The demonstration when applying Content-Based Transformer (CBT) to perform the sequential recommendation task, where each item is categorized into a distinct group indicated by three different colors. The red arrow represents the drift point of the user preference, which can be indicated when the item's group changes from one to another. The model establishes the relations among items from the same group and ignores irrelevant information as items from the other group.

The remainder of this paper is organized as follows: Section II provides further details of related work and Section III presents the proposed method. Section IV provides the experiment settings and evaluation results. The discussion of the results are presented in Section V, and we conclude our work in Section VI.

II. RELATED WORK

This section reviews the Sequential Recommender System (SRS) and the essential mechanism of Content-Based Transformer (CBT) [29], [30]. We further discuss the CBT for sequential recommendation task. Then, we present the differences between our proposed model and related works.

A. SEQUENTIAL RECOMMENDER SYSTEM

SRS is proposed to model the dynamics of user preference in a real-world scenario and recommend items relevant to the current user preference. Early works leverage Recurrent Neural Network (RNN) such as LSTM [31] and GRU [32] to consider the historical sequence of the user and deliver the next item prediction [6], [14], [24], [33]–[36]. Besides, other architectures of neural networks are also adopted,

such as Convolutional Neural Networks [37]–[39], Graph Neural Networks [40]–[42], and Transformer [17], [18], [21], [43], [44].

Nonetheless, the mentioned works consider every interaction in the historical sequence, which contain noises from irrelevant interactions. DSAN [19] is proposed to focus only on interactions with high relevance to the current user preference. The model leverages the self-attention mechanism to establish relations among interactions in the historical sequence. When computing the attention score, it ignores less relevant interactions by leveraging a sparse activation function [22]. Mathematically, the expression for computing the attention score between interactions ($\mathbf{H}$) in the self-attention is:

$$\mathbf{Q} = \text{ReLU}\left(\mathbf{W}^{Q}\mathbf{H} + \mathbf{b}^{Q}\right), \tag{1}$$

$$\mathbf{K} = \mathbf{V} = \mathbf{H}, \tag{2}$$

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \eta - \text{entmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{2d}}\right)\mathbf{V}, \tag{3}$$

$$\eta = \sigma\left(\mathbf{W}^{\eta}\mathbf{h}_{CLS} + \mathbf{b}^{\eta}\right) + 1, \tag{4}$$

where $\mathbf{W}^{Q}$ and $\mathbf{W}^{\eta}$ are learnable projection matrices. $\mathbf{b}^{Q}$ and $\mathbf{b}^{\eta}$ are learnable biases. d is the dimension of the latent representation, and σ is the sigmoid activation function. $\mathbf{h}_{CLS}$ is the latent representation of the CLS token, which contains information about the historical sequence's length and is utilized as the sequence representation. η is the magnitude of sparsity for entmax function.

However, DSAN determines the magnitude of sparsity (η) by the length of the historical sequence rather than deriving it based on the current user preference. DSAN also does not model user preference dynamics by capturing the drift of user preference. Hence, the model cannot capture the changes in the user preferences and may recommend irrelevant items.

B. CONTENT-BASED TRANSFORMER ARCHITECTURE

Numerous works proposed to modify the attention mechanism in the Transformer [28]–[30], [45]–[49]. One line of work aim to reduce the computational complexity when performing the self-attention [28], [45]–[49]. The key idea is to reduce the computational cost of $\mathcal{O}(l^2)$ due to the self-attention [16], which expresses as follows:

$$\mathbf{Q} = \mathbf{W}^{Q}\mathbf{H}, \quad \mathbf{K} = \mathbf{W}^{K}\mathbf{H}, \ \mathbf{V} = \mathbf{W}^{V}\mathbf{H}, \tag{5}$$

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{d}}\right)\mathbf{V}, \tag{6}$$

where l is the sequence's length, $\mathbf{H} \in \mathbb{R}^{d\times l}$ is the stacked item representations, and d is the dimension of item embedding dimension. $\mathbf{W}^{Q}$, $\mathbf{W}^{K}$, and $\mathbf{W}^{V} \in \mathbb{R}^{d\times d}$ are learnable projection matrices.

One possible solution is to utilize a sparse attention pattern instead of computing dense attention values between each pair of interactions in the sequence. Various works proposed different pre-defined attention patterns to reduce the computation complexity when considering long sequences [28], [45], [50]. For example, random patterns in Big Bird [45], sliding windows and dilated sliding windows in Longformer [28]. However, applying these works for sequential recommender tasks will consider past interactions regardless of current user preference. Moreover, the attention pattern is not personalized, which causes the model to consider interactions that may not be relevant to the current user preference.

Another possible approach is to compute the whole attention map while reducing the computation complexity via kernel [46] or reordering the computation of self-attention [47]. Although [46] and [47] can reduce the complexity of the Transformer from $\mathcal{O}(l^2)$ to $\mathcal{O}(l)$, applying these models for the SRS task will result in considering the whole historical sequence. As a result, the model's input will contain noises from interactions irrelevant to the current user preference. Hence, the linear-complexity Transformers [46], [47] are not related to our approach that considers only past interactions relevant to the current user preference.

Content-Based Transformer (CBT) is another variant of the Transformer, where each item in the historical sequence establishes the relations only with items having similar characteristics [29], [30]. The attention pattern of CBT is dynamic and based on the item characteristics in the sequence rather than utilizing the pre-defined attention pattern [28], [45], [49]. The demonstration when applying CBT to perform the sequential recommendation task is shown in Fig. 2, where each item is categorized into a different group. Applying CBT to perform the sequential recommendation task will result in grouping interactions with a similar user preference since CBT allows only interactions with similar characteristics to interact with each other. Consequently, CBT can indicate the drift of the user preference as the point where the item's group changes. The illustration of drift points is depicted in Fig. 2 as the red arrow. To deliver the prediction relevant to the current user preference, CBT can utilize the latest interaction as a reference point to retrieve only the relevant past interactions. Therefore, CBT ignores irrelevant past interactions to the current user preference, which reduces the noises from the model's input.

We will review related CBTs in the following subsections, namely Reformer [29] and Routing Transformer [30], which have different methods to compute the similarity between items. Note that the word "group" and "cluster" may be used interchangeably, depending on the method of each model for computing similarity between items.

1) REFORMER

This model employs Local Sensitive Hash (LSH) to compute the hash value (b) for categorizing each item into its corresponding group. Mathematically, the computation of the LSH can be expressed as follows:

$$\text{LSH}(\mathbf{h}) = \text{arg-max}\ ([\mathbf{h}_t\mathbf{R}; -\mathbf{h}_t\mathbf{R}]), \tag{7}$$

where $\mathbf{h}_t$ is the item representation, $\mathbf{R} \in \mathbb{R}^{d\times(b/2)}$ is the random matrix, and $[\cdot;\cdot]$ is the concatenation of vectors. As a result, similar items will have a high probability of getting

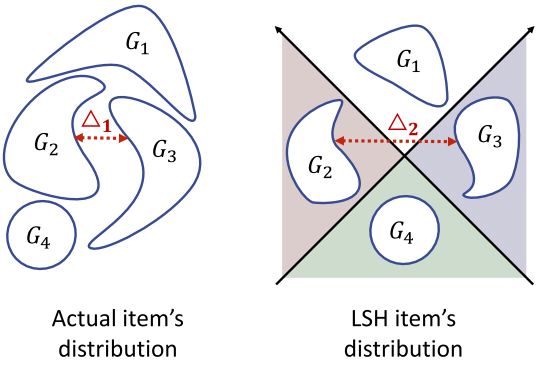


FIGURE 3. The demonstration of LSH's limitations, given four groups of items. The items' distributions in LSH differ from the actual distributions due to fixed categorized conditions. The item characteristics are inaccurate since the distances between groups of items are different. The distance between G_2 and G_4 in the actual distribution ($\triangle_1$) is lower than the distance in the LSH ($\triangle_2$).

the same hash value and being classified into the same group. When computing the attention score, Reformer allows only items in the same group to attend with each other:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}, \quad \text{where} \quad \text{LSH}(\mathbf{q}) = \text{LSH}(\mathbf{k}). \tag{8}$$

Additionally, the key ($\mathbf{k}$) is equal to the query ($\mathbf{q}$) with a unit norm, which ensures that none of the groups will contain only queries ($\mathbf{Q}$) or keys ($\mathbf{K}$):

$$\mathbf{k}_t = \frac{\mathbf{q}_t}{\|\mathbf{q}_t\|}, \tag{9}$$

where $\mathbf{q}_t$ and $\mathbf{k}_t$ are query and key at t^{th} step, respectively. Despite the effectiveness of Reformer, the model optimizes only the item representations but not the condition for the item categorization. The fixed grouping condition may limit the model from optimizing item representations effectively, such that item representations may not represent the actual item characteristics. The illustration of LHS's limitation is shown in Fig. 3, given four groups of items (G_1 to G_4) and four regions in LSH. The distributions of G_1, G_2, and G_3 in LSH are different from their actual distributions due to the fixed categorize condition. Hence, the item characteristics are incorrectly expressed since the distance between G_2 and G_3 in LSH ($\triangle_2$) is higher than the distance from the actual distribution ($\triangle_1$).

In contrast, our proposed method is more flexible since it optimizes both item representations and grouping conditions as the clustering algorithm. As a result, item representations of our proposed method can express the actual item characteristics, allowing the model to capture user preference correctly for delivering relevant recommendations.

2) ROUTING TRANSFORMER

Routing Transformer leverages the k-Mean clustering algorithm to group similar items and allows only items in the same cluster to attend with each other. Similar to other Transformer models, Routing Transformer projects item representations

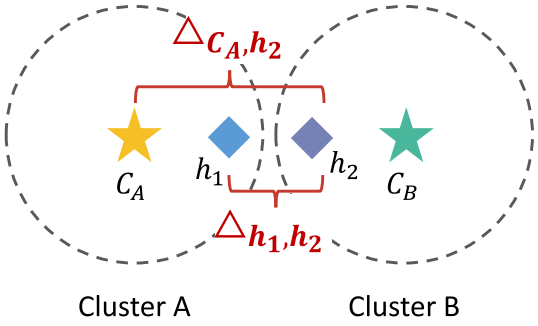


FIGURE 4. The illustration shows the limitation of clustering via the k-Mean algorithm. Although items h_1 and h_2 are similar, the k-Mean may assign these items into different clusters, resulting in an incorrect interpretation of the item characteristics.

to output queries ($\mathbf{Q}$), keys ($\mathbf{K}$), and values ($\mathbf{V}$). The model normalizes queries and keys before computing the attention map by the Layer Normalization (LN) [51]:

$$\mathbf{Q} = \text{LN}(\mathbf{Q}), \tag{10}$$
$$\mathbf{K} = \text{LN}(\mathbf{K}). \tag{11}$$

Then, Routing Transformer assigns each query and key to their corresponding cluster based on the similarity between centroids $\mathbf{C} \in \mathbb{R}^{|centroids| \times d}$:

$$\mathbf{Q}_{prod} = \mathbf{C}\mathbf{Q}^\top, \tag{12}$$
$$\mathbf{Q}_{idx} = \text{top-k}(\mathbf{Q}_{prod}, w_{size}), \tag{13}$$
$$\mathbf{Q}_{idx} = \text{sort}(\mathbf{Q}_{idx}), \tag{14}$$
$$\mathbf{K}_{prod} = \mathbf{C}\mathbf{K}^\top, \tag{15}$$
$$\mathbf{K}_{idx} = \text{top-k}(\mathbf{K}_{prod}, w_{size}), \tag{16}$$
$$\mathbf{K}_{idx} = \text{sort}(\mathbf{K}_{idx}), \tag{17}$$

where w_{size} is the number of keys to which the query can be attended, and sort is an operation to sort for maintaining the temporal structure of the sequence after the top-k function. The self-attention in Routing Transformer is computed by allowing the query to attend with only keys that are categorized into the same cluster:

$$\mathbf{Q}' = \text{gather}(\mathbf{Q}, \mathbf{Q}_{idx}), \tag{18}$$
$$\mathbf{K}' = \text{gather}(\mathbf{K}, \mathbf{K}_{idx}), \tag{19}$$
$$\mathbf{V}' = \text{gather}(\mathbf{V}, \mathbf{K}_{idx}), \tag{20}$$
$$\text{Attention}(\mathbf{Q}', \mathbf{K}', \mathbf{V}') = \text{Softmax}\left(\frac{\mathbf{Q}'\mathbf{K}'^\top}{\sqrt{d}}\right)\mathbf{V}', \tag{21}$$

where gather is a function to retrieve vectors based on the given indices. Rather than allow the model to optimize only the item representations as in Reformer, Routing Transformer updates the centroids via an online learning approach:

$$\mathbf{Q}_m = \text{one-hot}(\text{arg-max}(\mathbf{Q}_{prod})), \tag{22}$$
$$\mathbf{K}_m = \text{one-hot}(\text{arg-max}(\mathbf{K}_{prod})), \tag{23}$$
$$\mathbf{C} = \lambda\mathbf{C} + \frac{(1-\lambda)}{2}\mathbf{Q}_m\mathbf{Q} + \frac{(1-\lambda)}{2}\mathbf{K}_m\mathbf{K}, \tag{24}$$

TABLE 1. Comparison of the characteristics of related works and PPD+.

Characteristics	Longformer [28]	Reformer [29]	Routing Transformer [30]	DSAN [19]	PPD+
Hard clustering	-	✓	✓	-	✗
Soft clustering	-	✗	✗	-	✓
Not fixed cluster centroids	-	✗	✓	-	✓
Detection of user preference drift	✗	✓	✓	✗	✓
Items comparison for determining preference drift	-	Cluster	Cluster	-	Consecutive items
Items selection regarding current user preference	✗	✓	✓	✓	✓
Items selection as a period	✗	✓	✓	✗	✓

where λ is a decay value for the exponentially weighted average to update the centroids, and one-hot is a function to convert the integer to its one-hot representation. The loss function of the Routing Transformer consists of loss function from prediction and clustering:

$$\mathcal{L}_{total} = \mathcal{L}_{predict} + \alpha \mathcal{L}_{cluster}, \tag{25}$$

where the hyper-parameter α controls the influence of cluster loss on the total loss.

One advantage of Routing Transformer is employing a clustering algorithm, which allows the model to optimize item representations and centroids rather than utilizing fixed random vectors to categorize the item's group as LSH in Reformer. When adopting the Routing Transformer for the sequential recommendation task, the drift of user preference is represented by altering the item's cluster from one to another.

However, the clustering algorithm adopted in Routing Transformer provides a hard label to the item representations, which may not interpret the actual characteristics of the item in a real-world scenario. For instance, most movies cannot be categorized into one cluster since they contain numerous genres, such as romance, comedy, and action. Therefore, Routing Transformer may not capture the actual drift of user preference and result in false-positive drift signals since the k-Mean may categorize two items into different clusters, although their characteristics are similar. The limitation of the k-Mean algorithm is demonstrated in Fig. 4. Given two clusters and two items in the latent space, both item characteristics are similar since the distance between the two items ($\triangle_{h_1,h_2}$) is small. Nonetheless, the distance between cluster A and item h_2 ($\triangle_{C_A,h_2}$) is large, causing the h_2 to be categorized into cluster B, although h_1 and h_2 are similar.

On the other hand, our proposed method leverages the Fuzzy c-Mean algorithm to output a soft label for the item's cluster. The soft label corresponds to the actual item characteristics in a real-world scenario where items exist in many clusters. It further allows item representations to be flexibly present in multiple clusters and establishes relations with similar items in another cluster.

We summarize the comparison between our proposed method (PPD+) with related works and present it in Table 1. We compare every model in seven aspects: hard clustering, soft clustering, not fixed cluster centroids, detection of user preference drift, items comparison for determining preference drift, items selection regarding current user preference, and items selection as a period. In the first aspect, only Content-Based Transformers leverage the hard clustering, i.e., k-Mean in Routing Transformer and LSH in Reformer. On the other hand, PPD+ is the only model that utilizes soft clustering via Fuzzy c-Mean to group items with similar characteristics. Among models that employ clustering algorithms, Reformer is the only model that fixed the position of centroids. In contrast, other models allow the centroids' position to be changed and optimized. For the fourth aspect, Routing Transformer, Reformer, and PPD+ are the only models that detect user preference drift. The detection of user preference drift in Content-Based Transformers can be viewed as changing item groups. In comparison, PPD+ determines the drift by comparing consecutive item characteristics. Longformer is the only model that does not regard the current user preference due to the pre-defined attention pattern when considering historical sequence. Lastly, Routing Transformer, Reformer, and PPD+ retrieve past interaction as periods of similar items. On the other hand, DSAN and Longformer may retrieve past interactions as several separate interactions due to sparse activation function in DSAN and dilated sliding window in Longformer.

To this end, none of the related works can detect the actual drift of user preference to capture user preference dynamics for delivering relevant recommendations. Longformer and DSAN ignore user preference drift when delivering recommendations. Furthermore, Longformer ignores the current user preference when considering past interactions. The recommendation from these models may not be relevant to the current user preference since the attention pattern in Longformer and the degree of sparsity in DSAN are not determined based on the current user preference. In the case of Content-Based Transformers, the drift can be indicated by the change of item's group between consecutive items. Nonetheless, hard clustering can assign different groups to similar items and result in a false-positive drift signal. Furthermore, hard clustering cannot model item characteristics in a real-world scenario, where an item can belong to various groups simultaneously.

III. PROPOSED METHODS

Fig. 5 (a) illustrates an overview of our proposed model called PPD+. The model consists of three main components while training in an end-to-end approach with joint loss functions. We summarize each part of the proposed method as the algorithm in Algorithm 1, 2, and 3, respectively. In the first stage,

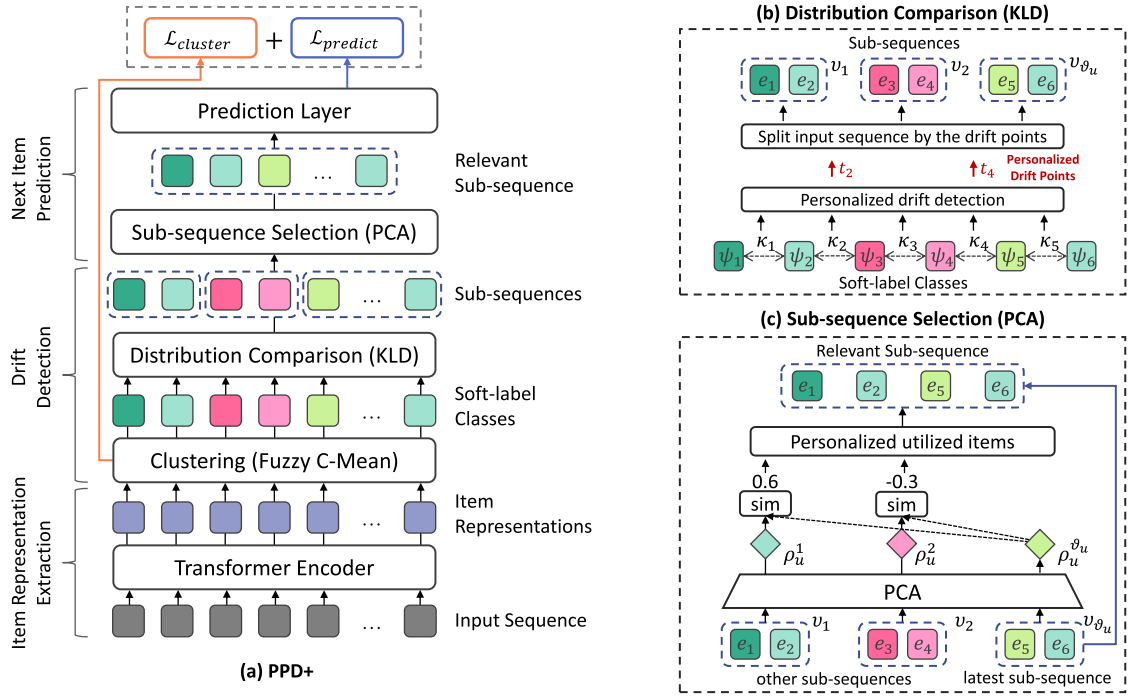


FIGURE 5. The illustration of (a) the proposed PPD+ consists of three main components with the combined loss function. The figure further depicts the methodology in (b) the distribution comparison and (c) the sub-sequences selection.

the model considers the historical sequence of the target user by employing the Transformer to extract contextualized item representations. Then, the drift detection in the second stage processes the item representations from the first stage to establish the personalized drift pattern of the user preference. The detected drift pattern is leveraged to retrieve past interactions relevant to the current user preference, which reduce the noises to the model's input from the irrelevant interactions.

In this work, the model utilizes the optimal number of items in the historical sequence by selecting only sub-sequences having a similarity with the current user preference greater than zero. The optimal item utilization allows PPD+ to overcome the limitation of our previous work [25] that required a hyper-parameter to control the utilization of the past items. The last stage retrieves the relevant past interactions as inputs and passes them to the prediction layer for delivering the next item prediction.

A. ITEM REPRESENTATIONS EXTRACTION

PPD+ adopts the Transformer Encoder [16] to compute the contextualized latent's representation of items in the historical sequence $\mathbf{S}_u$ of the target user u. PPD+ converts t^{th} item in the $\mathbf{S}_u$ into the item embedding $\mathbf{e}_t \in \mathbb{R}^d$ with associated positional embedding $\mathbf{p}_t \in \mathbb{R}^d$. The model combines both vectors as a latent vector $\mathbf{h}_t \in \mathbb{R}^d$, representing the item characteristics and temporal information:

$$\mathbf{h}_t = \mathbf{e}_t + \mathbf{p}_t. \tag{26}$$

To encode the contextualized information of other items present in the sequence into the latent vector ($\mathbf{h}_t$), the model utilizes the self-attention in the Transformer to establish relations among items in the sequence. The self-attention projects the latent vector ($\mathbf{h}_t$) into query ($\mathbf{q}_t$), key ($\mathbf{k}_t$), and value ($\mathbf{v}_t$):

$$\mathbf{q}_t = \mathbf{W}^Q\mathbf{h}_t, \quad \mathbf{k}_t = \mathbf{W}^K\mathbf{h}_t, \ \mathbf{v}_t = \mathbf{W}^V\mathbf{h}_t, \tag{27}$$

where $\mathbf{W}^Q$, $\mathbf{W}^K$, and $\mathbf{W}^V \in \mathbb{R}^{d\times d}$ are learnable projection matrices. Then, self-attention establishes relations between items by computing attention scores on stacked queries ($\mathbf{Q}$), keys ($\mathbf{K}$), and values ($\mathbf{V}$) of items in the $\mathbf{S}_u$:

$$\mathbf{H} = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}, \tag{28}$$

where $\sqrt{d}$ is required for training stability [16].

The Transformer introduces a non-linear transformation to the stacked latent representations ($\mathbf{H}$) by passing it into the series of Positional-wise Feed-Forward Network (PFFN), Dropout, Residual Connection, and Layer Normalization (LN):

$$\mathbf{H} = \text{LN}(\text{Dropout}(\text{PFFN}(\mathbf{H})) + \mathbf{H}), \tag{29}$$

$$\text{PFFN}(\mathbf{H}) = [\text{FFN}(\mathbf{h}_1); \ldots; \text{FFN}(\mathbf{h}_t)], \tag{30}$$

$$\text{FFN}(\mathbf{h}_t) = \text{ReLU}(\mathbf{W}_1\mathbf{h}_t + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2, \tag{31}$$

where $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d\times d}$ and $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^d$ are learnable weights and biases, respectively. The final item representations are passed as an input for the next component to detect the user preference drift.

B. DRIFT DETECTION

In this component, the model detects the personalized drift pattern of the user preference based on the user's

Algorithm 1 PPD+: Item Representations Extraction

Input: $\mathbf{S}_u$
Output: $\mathbf{H}$

Randomly initialized:
- $\mathbf{E} \in \mathbb{R}^{|item| \times d}$: Item embedding matrix
- $\mathbf{P} \in \mathbb{R}^{|\mathbf{S}_u| \times d}$: Positional embedding matrix
- $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d \times d}$: Projection matrices

for t=1 to $|\mathbf{S}_u|$ **do**
  $\mathbf{e}_t \leftarrow$ emb_lookup(i_t), $\mathbf{p}_t \leftarrow$ emb_lookup(t)
  $\mathbf{h}_t \leftarrow \mathbf{e}_t + \mathbf{p}_t$
end for
$\mathbf{Q} \leftarrow \mathbf{W}^Q\mathbf{H}, \mathbf{K} \leftarrow \mathbf{W}^K\mathbf{H}, \mathbf{V} \leftarrow \mathbf{W}^V\mathbf{H}$
$\mathbf{H} \leftarrow \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V}$
$\mathbf{H} \leftarrow \text{LayerNorm}(\text{Dropout}(\text{PFFN}(\mathbf{H})) + \mathbf{H})$

behavior encoded in the latent representations from the previous component. The drift point is the point in time when item characteristics are significantly different from the previous item. PPD+ utilizes a clustering algorithm to capture item characteristics, such that similar items will be categorized into the same cluster. The Fuzzy c-Mean algorithm is adopted as the clustering algorithm to correspond with item characteristics in a real-world scenario where an item can exist in various groups.

1) CLUSTERING THE LATENT REPRESENTATIONS

Given latent representations ($\mathbf{H}$) from the previous component, the Fuzzy c-Mean categorizes the representation based on the distance between centroids. It outputs the soft label classes ($\boldsymbol{\psi}$) as the probability of classifying the item (h) into each cluster (c):

$$\boldsymbol{\psi}_{s,n} = \left[\sum_{r=1}^{NC}\left(\frac{\|\mathbf{h}_n - \mathbf{c}_{nc}\|}{\|\mathbf{h}_n - \mathbf{c}_r\|}\right)^{\frac{2}{\phi-1}}\right]^{-1}, \quad \forall n, \tag{32}$$

$$\mathbf{c}_{nc} = \frac{\sum_{n=1}^{N}\boldsymbol{\psi}_{nc,n}^{\phi}\mathbf{h}_n}{\sum_{n=1}^{N}\boldsymbol{\psi}_{nc,n}^{\phi}}, \quad \forall nc. \tag{33}$$

Here, NC is the number of clusters, n is the total number of latent representations, and $\phi \in (1, \infty)$ is the fuzzification constant. The goal of the Fuzzy c-Mean is to minimize the objective function:

$$\mathcal{L}_{cluster}(\boldsymbol{\Psi}, \mathbf{C}) = \sum_{n=1}^{|H|}\sum_{nc=1}^{NC}\boldsymbol{\psi}_{nc,n}^{\phi}\|\mathbf{h}_n - \mathbf{c}_{nc}\|^2. \tag{34}$$

Typically, the Fuzzy c-Mean will repeat the computation of (32) and (33) until the $\mathcal{L}_{cluster}$ is converged. However, PPD+ adopts the online update of (33) via the exponential moving average to enable the model to be trained in an end-to-end approach:

$$\mathbf{C}_{new} = \lambda\mathbf{C}_{new} + (1-\lambda)\mathbf{C}_{old}, \tag{35}$$

where λ is the decay value of the exponentially weighted average.

Algorithm 2 PPD+: Drift Detection

Input: $\mathbf{H}, \boldsymbol{\phi}$
Output: $\mathbf{S}'_u$

Randomly initialized:
- $\mathbf{C} \in \mathbb{R}^{NC \times d}$: Centroids

▽ **FUZZY C-MEAN**
for $n = 1$ to $|\mathbf{H}|$ **do**
  $\boldsymbol{\psi}_{s,n} \leftarrow \left[\sum_{r=1}^{NC}\left(\frac{\|\mathbf{h}_n - \mathbf{c}_{nc}\|}{\|\mathbf{h}_n - \mathbf{c}_r\|}\right)^{\frac{2}{\phi-1}}\right]^{-1}$
end for
for $nc = 1$ to NC **do**
  $\mathbf{c}_{nc} \leftarrow \frac{\sum_{n=1}^{N}\boldsymbol{\psi}_{nc,n}^{\phi}h_n}{\sum_{n=1}^{N}\boldsymbol{\psi}_{nc,n}^{\phi}}$
end for
$\mathbf{C}_{new} \leftarrow \lambda\mathbf{C}_{new} + (1-\lambda)\mathbf{C}_{old}$
▽ **DISTRIBUTION COMPARISON**
for $t = 1$ to $|\mathbf{S}_u| - 1$ **do**
  $\kappa_t \leftarrow \sum_{nc=1}^{NC}\boldsymbol{\psi}_{nc}^{t}\log\left(\frac{\boldsymbol{\psi}_{nc}^{t}}{\boldsymbol{\psi}_{nc}^{t-1}}\right)$
end for
$\mu_u \leftarrow \frac{\sum\kappa}{|\kappa|}$
$\sigma_u \leftarrow \sqrt{\frac{(\kappa_t - \mu_u)^2}{|\kappa|-1}}$
drift_points $= \{0\} \cup \{t | \kappa_t > \mu_u + \sigma_u\} \cup \{|\mathbf{S}_u|\}$
for $j = 1$ to $|$**drift_points**$|$ **do**
  $\boldsymbol{\upsilon}_u^j \leftarrow \mathbf{S}_u[$**drift_points**$[j] :$ **drift_points**$[j+1]]$
end for
$\mathbf{S}'_u \leftarrow \{\boldsymbol{\upsilon}_u^j, \cdots, \boldsymbol{\upsilon}_u^{\vartheta_u}\}$

2) DISTRIBUTION COMPARISON

To detect the drift of the user preference, the model compares consecutive item characteristics retrieved from the Fuzzy c-Mean in the form of soft label classes ($\boldsymbol{\psi}$), as shown in Fig. 5 (b). It employs the KL-Divergence (KLD) to compare consecutive soft label classes since their values are the probability distribution:

$$\kappa_t = D_{KL}(\boldsymbol{\psi}^t || \boldsymbol{\psi}^{t-1}) = \sum_{nc=1}^{NC}\boldsymbol{\psi}_{nc}^{t}\log\left(\frac{\boldsymbol{\psi}_{nc}^{t}}{\boldsymbol{\psi}_{nc}^{t-1}}\right). \tag{36}$$

The model indicates the drift point of the user preference as the point where the κ_t is peak and higher than the personalized threshold. It computes the mean (μ_u) and standard deviation (σ_u) of κ for each user u and define $\mu_u + \sigma_u$ as the threshold. Since different users have different past behaviors, the threshold of κ and the detected drift pattern of the user preference are personalized.

PPD+ leverages the detected drift points to divide the historical sequence into multiple sub-sequences $\mathbf{S}'_u = [\boldsymbol{\upsilon}_u^1, \ldots, \boldsymbol{\upsilon}_u^{\vartheta_u}]$, such that each sub-sequence ($\boldsymbol{\upsilon}_u$) contains only a particular user preference.

C. NEXT ITEM PREDICTION

To deliver the recommendation relevant to the current user preference, the model considers only the past interactions containing similar preferences to the current user preference. The other past interactions are ignored to reduce

Algorithm 3 PPD+: Next Item Prediction

Input: $\mathbf{S}'_u$, $\mathbf{E}$, $\mathbf{P}$
Output: $\hat{\mathbf{y}}$

Randomly initialized:
- $\mathbf{W}^{Q'}, \mathbf{W}^{K'}, \mathbf{W}^{V'} \in \mathbb{R}^{d\times d}$: Projection matrices
- $\mathbf{W}_3 \in \mathbb{R}^{|item|\times d}$: Projection matrix
- $\mathbf{b}_3 \in \mathbb{R}^{|item|}$: Bias

▽ **SUB-SEQUENCE SELECTION**

for $j = 1$ to ϑ_u **do**
  $\mathbf{HP} \leftarrow \{\mathbf{h}|\mathbf{h} \in \upsilon_u^j\}$
  $\bar{\mathbf{HP}} \leftarrow \frac{1}{|\mathbf{HP}|}\sum_{g=1}^{d}\mathbf{HP}_{:,g}$
  $\tilde{\mathbf{HP}} \leftarrow \mathbf{HP} - \bar{\mathbf{HP}}$
  $\mathbf{COV} \leftarrow \tilde{\mathbf{HP}}^{\top}\tilde{\mathbf{HP}}$
  $\boldsymbol{\rho}_u^j \leftarrow$ eigenvector with a highest eigenvalue of **COV**
end for
for $j = 1$ to ϑ_u **do**
  $sim(\boldsymbol{\rho}_u^j, \boldsymbol{\rho}_u^{\vartheta_u}) \leftarrow \frac{\boldsymbol{\rho}_u^j \cdot \boldsymbol{\rho}_u^{\vartheta_u}}{\|\boldsymbol{\rho}_u^j\| \times \|\boldsymbol{\rho}_u^{\vartheta_u}\|}$
end for
$\mathbf{S}_u^{rel} = \left\{ i_t | i_t \in \boldsymbol{\upsilon}_u^j, sim(\boldsymbol{\rho}_u^j, \boldsymbol{\rho}_u^{\vartheta_u}) > 0 \right\}$

▽ **PREDICTION LAYER**

append *CLS* to $\mathbf{S}_u^{rel}$
for t=1 to $|\mathbf{S}_u^{rel}|$ **do**
  $\mathbf{e}_t \leftarrow emb\_lookup(i_t)$, $\mathbf{p}_t \leftarrow emb\_lookup(t)$
  $\mathbf{h}_t \leftarrow \mathbf{e}_t + \mathbf{p}_t$
end for
$\mathbf{Q} \leftarrow \mathbf{W}^{Q'}\mathbf{H}$, $\mathbf{K} \leftarrow \mathbf{W}^{K'}\mathbf{H}$, $\mathbf{V} \leftarrow \mathbf{W}^{V'}\mathbf{H}$
$\mathbf{H} \leftarrow \text{Softmax}\left(\frac{\mathbf{QK}^{\top}}{\sqrt{d}}\right)\mathbf{V}$
$\mathbf{H} \leftarrow \text{LayerNorm}(\text{Dropout}(\text{PFFN}(\mathbf{H})) + \mathbf{H})$
$\hat{\mathbf{y}} \leftarrow \text{Softmax}(\mathbf{W}_3\mathbf{h}_{CLS} + \mathbf{b}_3)$

noises as irrelevant information from the model's input. As a result, the model is required to compare past interactions with the current user preference to retrieve relevant interactions. Although the historical sequence is divided into many sub-sequences that contain only one user preference, the method for comparing sub-sequences with unequal numbers of items is not trivial. Hence, the model utilizes the Principal Component Analysis (PCA) to compute the representation of each sub-sequence in terms of distribution. It represents interactions in the sub-sequence by the principal component with the highest eigenvalues, as shown in Fig. 5 (c).

1) SUB-SEQUENCE SELECTION

In PCA, the latent representations $\mathbf{h}_{u,i}^t \in \mathbb{R}^d$ in each sub-sequence ($\boldsymbol{\upsilon}_u$) are stacked into a $\mathbf{HP} \in \mathbb{R}^{|\upsilon_u|\times d}$. Then, PCA normalizes the **HP** by subtracting each column with its mean values and computes the covariance matrix of the **HP** by multiplying the **HP** with its transpose:

$$\bar{\mathbf{HP}} = \frac{1}{|\mathbf{HP}|}\sum_{g=1}^{d}\mathbf{HP}_{:,g}, \tag{37}$$

$$\tilde{\mathbf{HP}} = \mathbf{HP} - \bar{\mathbf{HP}}, \tag{38}$$

$$\mathbf{COV} = \tilde{\mathbf{HP}}^{\top}\tilde{\mathbf{HP}}. \tag{39}$$

Note that the g^{th} column of the matrix **HP** is represented by $\mathbf{HP}_{:,g}$. Next, the PCA calculates eigenvectors ($\boldsymbol{\rho}$) with corresponding eigenvalues as principal axes. The model utilizes the eigenvector ($\boldsymbol{\rho}$) with the highest eigenvalues to represent the sub-sequence ($\boldsymbol{\upsilon}_u$) since it captures most of the items' distribution variance.

To retrieve sub-sequences with similar preferences to the current user preference, the model represents the current user preference by the latest sub-sequence and compares it with other sub-sequences. In particular, PPD+ utilizes the cosine similarity to compare the representation of the latest sub-sequence ($\boldsymbol{\rho}_u^{\vartheta_u}$) and other sub-sequences ($\boldsymbol{\rho}_u^j$):

$$sim(\boldsymbol{\rho}_u^j, \boldsymbol{\rho}_u^{\vartheta_u}) = \frac{\boldsymbol{\rho}_u^j \cdot \boldsymbol{\rho}_u^{\vartheta_u}}{\|\boldsymbol{\rho}_u^j\| \times \|\boldsymbol{\rho}_u^{\vartheta_u}\|}. \tag{40}$$

The value of cosine similarity interprets the angle between two vectors with a degree between 0 and 180, which result in the range of cosine similarity between -1 and 1. The similarity is -1 when both vectors are in the opposite direction (180°), while the cosine value of 0 shows that both vectors are orthogonal to each other (90°). In contrast, the cosine value of 1 indicates that both vectors are the same (0°). Since the negative cosine similarity value indicates that both vectors are in a different direction, the model selects only the sub-sequences with a cosine similarity greater than zero. The positive similarity value ensures that the selected sub-sequences are relevant to the current user preference. The model also includes the latest sub-sequence to the relevant sub-sequence ($\mathbf{S}_u^{rel}$) as information about the current user preference. Finally, the output is the relevant sub-sequence, containing the id of each item and passed to the prediction layer for the next item prediction.

$$\mathbf{S}_u^{rel} = \left\{ i_t | i_t \in \boldsymbol{\upsilon}_u^j, sim(\boldsymbol{\rho}_u^j, \boldsymbol{\rho}_u^{\vartheta_u}) > 0 \right\} \tag{41}$$

2) PREDICTION LAYER

In the prediction layer, we append a special CLS token to the relevant sub-sequence ($\mathbf{S}_u^{rel}$) and exploit it as the sequence's representation for the next item prediction task. Similar to Section III-A, the model leverages the Transformer to retrieve contextualized item latent representations in the relevant sub-sequence ($\mathbf{S}_u^{rel}$). The model utilizes the latent representation of the CLS ($\mathbf{h}_{CLS}$) for computing the probability distribution of each item as the next item for the given sequence:

$$\hat{\mathbf{y}} = \text{Softmax}(\mathbf{W}_3\mathbf{h}_{CLS} + \mathbf{b}_3), \tag{42}$$

where $\mathbf{W}_3 \in \mathbb{R}^{|item|\times d}$ and $\mathbf{b}_3 \in \mathbb{R}^{|item|}$ are learnable weight and bias, respectively. The next item prediction task is optimized by the negative log-likelihood loss, which compares the predicted distribution ($\hat{\mathbf{y}}$) with the ground truth distribution ($\mathbf{y}$):

$$\mathcal{L}_{predict}(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{j=1}^{|item|} -y_j \log \hat{y}_j. \tag{43}$$

To this end, the model consists of two loss functions: cluster loss and prediction loss from the drift detection and the next item prediction, respectively. Similar to Routing Transformer [30], we combine both loss functions, allowing the model to be jointly optimized and applicable to train end-to-end:

$$\mathcal{L}_{total} = \mathcal{L}_{predict} + \alpha \mathcal{L}_{cluster}, \tag{44}$$

where the hyper-parameter α controls the influence of cluster loss on the total loss.

The computational complexity of PPD+ is $\mathcal{O}((l^2 \times d) + (l \times NC \times d) + (l^2 \times d) + (|item| \times d))$, where l is the length of the historical sequence, d is the item embedding size, NC is the number of clusters, and $|item|$ is the number of items in the system. Similar to [16], the computational complexity when utilizing the Transformer Encoder grows quadratically with the length of the historical sequence, i.e., $\mathcal{O}(l^2 \times d)$. During clustering and drift detection, the computational complexity grows with the historical sequence's length and the latent representation dimension, i.e., $\mathcal{O}(l \times NC \times d)$. In contrast, the calculation of sub-sequence selection has a complexity that grows quadratically to the length of the historical sequence dominated by the PCA, i.e., $\mathcal{O}(l^2 \times d)$. When delivering the recommendation, the computation complexity scales with the number of items in the system since the model calculates the probability of every item as the next item for the given historical sequence, i.e., $\mathcal{O}(|item| \times d)$.

IV. EXPERIMENTS

In this section, we performed experiments to evaluate the effectiveness of the proposed method to train in an end-to-end approach and to provide item's group as a soft label rather than hard label as in Content-Based Transformer. Moreover, we compared PPD+ with other methods which do not consider the whole historical sequence to verify that PPD+ selects more relevant past interactions than the baselines. We compared the proposed method with the baselines as follows:

- **Reformer** [29]: This model utilizes the Local Sensitive Hash (LSH) to group items with similar characteristics and establishes relations among items in the same group.
- **Routing Transformer** [30]: This model adopts the k-Mean clustering algorithm to group similar items while optimizing the model for clustering and prediction tasks.
- **Longformer** [28]: This model utilizes a pre-defined sparse attention map based on the position of the item in the historical sequence. Two variants of this model are adopted where the difference is their attention pattern: Longformer-SW and Longformer-DW. The Longformer-SW uses a sliding window attention pattern to establish relations among adjacent neighbor items. In contrast, Longformer-DW employs dilated sliding windows to skip some neighbors, which increases the receptive field.

TABLE 2. The statistics of datasets before and after pre-processing steps.

Before pre-processing steps

Dataset	MovieLens	Goodreads	UserBehavior
#users	6,040	18,892	987,994
#items	3,706	25,475	4,162,024
#interactions	1,000,209	1,378,033	100,150,807
Avg. length	165.6	72.9	101.4

After pre-processing steps

Dataset	MovieLens	Goodreads	UserBehavior
#users	6,040	15,435	47,597
#items	3,260	25,180	15,508
#interactions	998,539	1,358,607	4,330,537
Avg. length	165.3	88.0	90.9

- **DSAN** [19]: This model considers only highly relevant interactions by leveraging the sparse activation function. DSAN reported superior results over previous state-of-the-art.

The rest of this section is organized as follows: we describe the details of each dataset and explain the experiment setup. Then, we present the evaluation results of every model.

A. DATASETS

To evaluate all the methods, we conducted the experiments on three public benchmark datasets with different domains as follows:

- **MovieLens**[1]: A benchmark dataset in the movie domain, consisting of approximately 1 million interactions from every user.
- **Goodreads**[2]: A collection of user interactions in Book domains [52], [53]. The Spoiler subset is adopted, which consists of multiple book categories.
- **UserBehavior**[3]: A dataset contains over 100 million user interactions from an online e-commerce (Taobao) [54]–[56].

To prepare the datasets for the next item recommendation task, we converted every interaction into implicit feedback. We created the historical sequence for each user by grouping the user's interacted items and sorted the sequence based on the time stamp to maintain the temporal structure. To achieve computational tractability for a large dataset, i.e., UserBehavior, we randomly sampled 50,000 users from the dataset.

Since we aim to model the dynamics of the user preference, the length of the historical sequence should be sufficient to contain several users' preferences. For every dataset, we removed cold-start items and users by omitting users and items with less than ten interactions. Following [17], we set the maximum length of the historical sequence to be the rough approximation of the average sequence length. As shown in Table 2, the average length of the historical sequence of MovieLens, Goodreads, and UserBehavior are 165.6, 72.9, and 101.4, respectively. Therefore, we set the

[1]https://grouplens.org/datasets/movielens/1m
[2]https://github.com/MengtingWan/goodreads
[3]https://tianchi.aliyun.com

maximum sequence length of MovieLens, Goodreads, and UserBehavior as 200, 100, and 100. The statistics of each dataset before and after the pre-processing steps are presented in Table 2.

B. TASK SETTING

To evaluate the effectiveness of the model for the next item prediction, we held out the last interacted item of each historical sequence as the ground truth label. Therefore, the task of the model becomes to rank every item as the next item for the given historical sequence. Following [37], we utilized the first 70% of interactions in the historical sequence as a training set, the next 10% as a validation set for hyper-parameters tuning, and the remaining 20% as a test set for evaluation.

C. EVALUATION METRICS

We employed three raking metrics to measure the quality of the ranking retrieved from the next item prediction task. The Normalized Discounted Cumulative Gain (NDCG) is the first metric to calculate the rank quality for top-K items. The NDCG measures the quality of items' order by assigning a higher weight to top-rank items. Specifically, the NDCG can be expressed as follows:

$$\text{DCG@}K = \sum_{j=1}^{K} \frac{2^{rel_j} - 1}{\log_2(j+1)}, \tag{45}$$

$$\text{IDCG@}K = \sum_{j=1}^{|\text{REL}_k|} \frac{2^{rel_j} - 1}{\log_2(j+1)}, \tag{46}$$

$$\text{NDCG@}K = \frac{\text{DCG@}K}{\text{IDCG@}K}, \tag{47}$$

where $|\text{REL}_k|$ is the list of relevance items sorted by its relevance score up to K^{th} position.

The Hit Ratio (HR) is the second metric to evaluate the ranking quality by measuring the number of times the predicted item presents in the top-K list:

$$\text{HR@}K = \frac{1}{K} \sum_{j=1}^{K} rel_j. \tag{48}$$

The Mean Reciprocal Rank (MRR) is the last metric to measure the quality of the ranking, expressed as follows:

$$\text{MRR} = \frac{1}{|S|} \sum_{j=1}^{|S|} \frac{1}{rank_j}, \tag{49}$$

where $rank_j$ is the rank of the predicted item on the ground-truth label.

In this work, we focused on top-ranked items and reported NDCG@K and HR@K with $K = \{1, 5, 10\}$ since the ground-truth label for each sequence is only one item.

D. IMPLEMENTATION DETAILS

For every model, we considered the dimension of the item embedding from {16, 32, 64, 128}. The learning rate of the model is considered from {1e-2, 1e-3, 1e-4} and the batch size from {128, 256, 512, 1024}.

In Reformer, we considered the number of LSH from {1, 2, 5, 10, 15}. Note that one LSH can divide items into two groups, such that the given number of LSH can be equally viewed as 2, 4, 10, 20, and 30 clusters. For Routing Transformer and PPD+, we selected the number of clusters from {2, 4, 10, 20, 30} and randomly initialized the centroids. The w_{size} in Routing Transformer is set to the length of the sequence for a fair comparison with other models. The hyper-parameter α to control the trade-off between the next item prediction loss and the clustering loss is set to 1e-4. The hyper-parameter λ to control the update of centroids via exponential moving average is set to 0.999. We set the fuzzification value (ϕ) of PPD+ to 2.

In the case of Longformer, the model can be further divided into two variants according to its attention pattern: Longformer-SW using a sliding window and Longformer-DW using a dilated sliding window. These models need a pre-defined window size to establish the attention pattern. On the one hand, the small window size will cause the model to have sufficient information for delivering recommendations. On the other hand, a large window size will result in unequal item utilization compared to other models. For a fair comparison with other models, we set the window size of Longformer-SW as 50% of the total sequence length, resulting in consideration of the last 50% of the historical sequence. Similarly, the dilated size of Longformer-DW is one, which results in item utilization of approximately 50% of the total sequence length. In particular, Longformer-DW will skip one item after considering one interaction.

For DSAN, we selected its normalized weight factor [19] from {1, 5, 10, 15, 20} and dropout rate from {10, 25, 50}. We employed the implementation of sparse activation function from Lingvo framework [57].

The weights of every model are randomly initialized and trained from scratch. We set the dropout rate of Transformer in every model as 10% to alleviate overfitting. We adopted the Adam [58] to optimize the weight of every model.

E. EXPERIMENTAL RESULTS

We present the value from searching the optimal hyper-parameters and select the value of the hyper-parameters based on the value of NDCG@10 on the validation set.

1) ITEM EMBEDDING DIMENSION SIZE d

We selected the value of the item embedding dimension size (d) from {16, 32, 64, 128}. Fig. 6 shows the results in terms of NDCG@10. Most models have an optimal embedding size of 128 in every dataset. We observe that the trend of NDCG@10 increases as the number of embedding sizes increases. This may be due to the higher degree of freedom provided by a large embedding size since it can express item characteristics more than a smaller embedding size.

However, some models may have an optimal item embedding size of 64, e.g., DSAN and Longformer-SW in

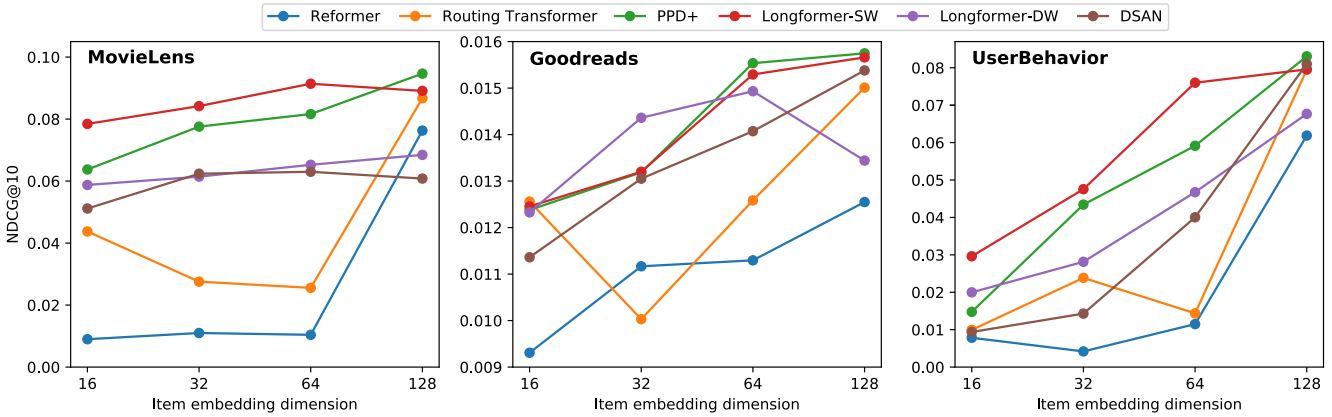


FIGURE 6. Impact of the number of item embedding size.

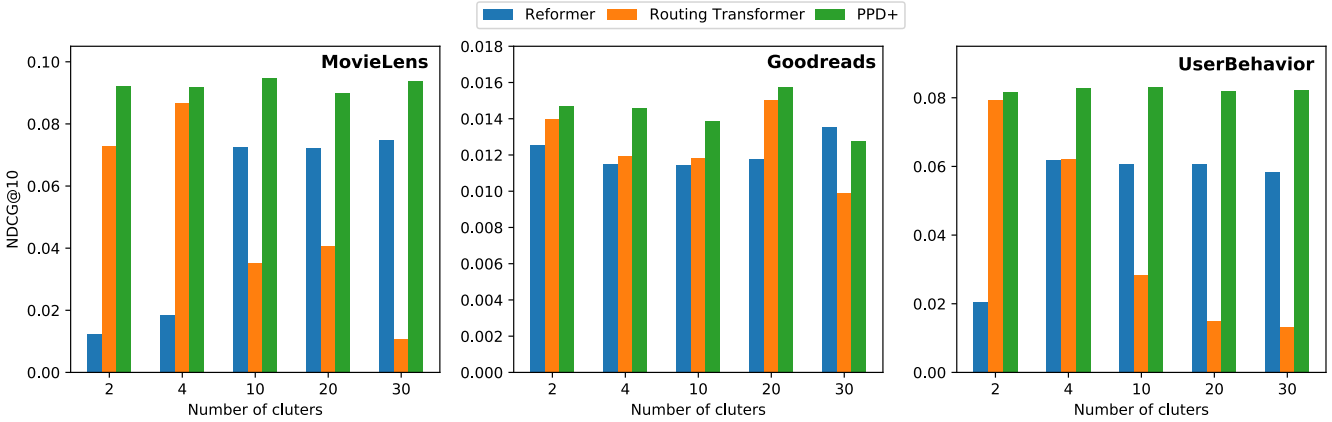


FIGURE 7. Impact of the number of clusters.

MovieLens and Longformer-DW in Goodreads. Overfitting may be a possible explanation for the drop in performance when increases the dimension of the item embedding.

2) NUMBER OF CLUSTERS *NC*

The results of the varying number of clusters (*NC*) are presented in Fig. 7, where we selected the value of *NC* from {2, 4, 10, 20, 30}. Note that only Reformer, Routing Transformer, and PPD+ have this hyper-parameter since these models leverage the clustering algorithm. In PPD+, the optimal values of *NC* are 10, 20, and 10 for MovieLens, Goodreads, and UserBehavior, respectively. Routing Transformer has the optimal value of *NC* as 4, 20, and 2 for MovieLens, Goodreads, and UserBehavior. On the other hand, Reformer requires the optimal *NC* of 30, 30, and 4 for MovieLens, Goodreads, and UserBehavior.

For every model, the Goodreads dataset requires higher *NC* compared to other datasets. One possible explanation may be due to the highest number of items in the Goodreads datasets, as shown in Table 2. The optimal *NC* for PPD+ and Routing Transformer is relatively low for the other two datasets. In contrast, Reformer also has the highest *NC* on the MovieLens dataset while requiring relatively low *NC* in the UserBehavior dataset.

3) EVALUATION RESULTS

Table 3 presents the evaluation results of the proposed method and baselines. We show the results in terms of MRR, NDCG@K, and HR@K, where K is {1, 5, 10}. Since the NDCG@1 equals HR@1, we omit the HR@1 and report only the NDCG@1.

Among the baselines, Reformer has the worst performance in Goodreads and UserBehavior dataset. The poor performance of Reformer demonstrates the limitation of the fixed LSH, which may not correctly represent the actual group of items. Routing Transformer outperforms Reformer in every dataset and most metrics. The higher performance of the Routing Transformer shows the benefits of its flexibility which allows the centroids of k-Mean to optimize along with the recommendation task. Surprisingly, Longformer-SW with sliding window outperforms Content-Based Transformer and its variant, i.e., Longformer-DW with dilated sliding window, across most datasets and matrices. The possible explanation is that Longformer-SW has more exposure to the latest

TABLE 3. Comparison of the evaluation results on benchmark datasets.

Datasets	Metrics	Reformer	Routing Transformer	Longformer-SW	Longformer-DW	DSAN	PPD+
MovieLens	NDCG@1	0.02533	0.02434	0.02301	0.01474	0.01540	**0.02566**
	NDCG@5	0.05069	0.05361	0.05324	0.03759	0.03319	**0.05425**
	NDCG@10	0.06229	0.07209	0.07054	0.05079	0.04585	**0.07292**
	HR@5	0.07550	0.08377	0.08361	0.06076	0.05166	**0.08411**
	HR@10	0.11142	0.14156	0.13758	0.10199	0.09106	**0.14205**
	MRR	0.05725	0.06387	0.06311	0.04900	0.04470	**0.06563**
Goodreads	NDCG@1	0.00246	0.00188	0.00279	0.00233	0.00227	**0.00305**
	NDCG@5	0.00553	0.00583	0.00696	0.00603	0.00615	**0.00755**
	NDCG@10	0.00806	0.00838	0.00952	0.00858	0.00933	**0.01024**
	HR@5	0.00855	0.00998	0.01140	0.00998	0.01037	**0.01212**
	HR@10	0.01646	0.01795	0.01937	0.01788	0.02028	**0.02060**
	MRR	0.00829	0.00879	0.01019	0.00928	0.00894	**0.01078**
UserBehavior	NDCG@1	0.03397	0.03935	0.02566	0.03698	**0.04414**	0.04288
	NDCG@5	0.05042	0.06221	0.05532	0.05308	0.06542	**0.06560**
	NDCG@10	0.05560	0.07018	0.07259	0.05864	0.07270	**0.07344**
	HR@5	0.06563	0.08339	0.08493	0.06801	0.08526	**0.08629**
	HR@10	0.08168	0.10816	**0.13858**	0.08526	0.10786	0.11064
	MRR	0.05089	0.06392	0.06605	0.05483	0.06719	**0.06737**

interaction than Longformer-DW since it utilizes 50% of the latest interactions. Consequently, Longformer-DW has a higher chance of including interactions at the beginning of the historical sequence, which may not be relevant to the current user preference. In comparison, DSAN mostly outperforms other baseline models in Goodreads and UserBehavior datasets, whereas it has the worst performance in the MovieLens dataset.

When comparing PPD+ with the baselines, our proposed method consistently outperforms most baselines in terms of NDCG, HR, and MRR. The results illustrate the effectiveness of our proposed method as an end-to-end model to group the similar items by the Fuzzy c-Mean instead of fixed LSH in Reformer or k-Mean in Routing Transformer. Moreover, item utilization of PPD+ is regarding the current user preference, rather than a pre-defined pattern in Longformer or length of the historical sequence as in DSAN.

V. DISCUSSION

This section discusses the difference between our proposed method and the baselines. We analyze the impact of the number of the cluster on the performance of every model. Next, we discuss the differences between our proposed model with the baselines in terms of the similarity function. We verify the effectiveness of the proposed method for grouping similar items by conducting an additional experiment where each model utilizes an equal number of items. Then, we discuss the impact of noises in the input on every model's performance.

To demonstrate the behavior of each model under different circumstances, we present four case studies and discuss the impact on the model's prediction. In particular, the four cases are irrelevant latest interaction, incrementally changed user preference, incorrectly grouping similar items, and impact of different clustering algorithms.

Additionally, we discuss the impact of optimal item utilization on the model's performance. We present an additional experiment to demonstrate the competence of the proposed threshold by comparing it with variant models using different thresholds.

A. COMPARE WITH THE BASELINES

1) NUMBER OF CLUSTERS *NC*

According to Fig. 7, the performance of every model is affected by the number of clusters (*NC*). To investigate the impact of the *NC*, we analyze the number of utilized items as the percentage of utilized items since the sequence's length of each dataset is unequal, as shown in Table 2. Since different historical sequences have different item utilization, we show the percentage of utilized items as the average values. We present the analysis results in Fig. 8, where the x-axis is the number of clusters, and the y-axis is the average utilized items percentage.

Among the baseline models, we observe that Routing Transformer has relatively more utilized items than Reformer. The higher item utilization in Routing Transformer may result from its flexibility that allows the clustering algorithm to optimize along with the recommendation task. In contrast, Reformer optimizes only the item representations but not the grouping condition, which may not be an optimal condition for retrieving similar items. Reformer's limitation is demonstrated in Fig. 3, where the distance between the distribution of similar items differs from the true distribution. We also notice that the baselines have decreased the average percentage of utilized items when the *NC* increases. One possible explanation is that the baselines establish relations only among items belonging to the same cluster. The *NC* limits item utilization in the baselines since the number of items per cluster will decrease when the *NC* increases. Consequently, the baselines may not have sufficient past information to deliver the recommendations.

On the other hand, item utilization in PPD+ is not limited by the *NC*. PPD+ overcomes the baselines limitation by retrieving every past interaction similar to the current user preference. One advantage is that the percentage of utilized items is optimal and personalized based on the user's past

FIGURE 8. The percentage of utilized items in the sequence.

behavior in the historical sequence, resulting in more relevant recommendations.

2) GROUPING OF SIMILAR PAST INTERACTIONS

To verify the effectiveness of the proposed method to group and select past interactions without the impact from the percentage of utilized items, we performed an additional experiment by modifying every model to consider an equal amount of items in the historical sequence. For MovieLens, Goodreads, and UserBehavior, we set the percentage of utilized items to 30, 30, and 40, respectively. We followed the same experimental setting as in Section IV. Table 4 reports the evaluation results on the validation set of each model.

When comparing among the baselines, Routing Transformer outperforms Reformer in MovieLens and UserBehavior across most metrics. Reformer surpasses Routing Transformer in the Goodreads dataset. On the other hand, PPD+ outperforms the baselines in most metrics across every dataset. Since every model utilized the same amount of items, the superior evaluation results of PPD+ show that it can better group and select past interactions than the baselines to deliver the recommendations.

One possible explanation is that the baselines group similar items via the hard label method. They ignore the actual item characteristics in a real-world scenario where an item can be in many groups, e.g., a movie with various genres. Therefore, the baseline models may separate similar items into different groups and retrieve items in the same group, which may be irrelevant to the current user preference.

In contrast, PPD+ has more flexibility when grouping items since it utilizes the Fuzzy c-Mean to group similar items, where each item can belong to various clusters simultaneously. Grouping similar items via the Fuzzy c-Mean corresponds to characteristics of items in a real-world scenario where an item can be present in various groups. PPD+ can retrieve past interactions relevant to the current user preference, although each item may belong to different groups.

TABLE 4. Comparison of the evaluation results when the percentage of utilized items are equal.

	Metrics	Reformer	Routing Transformer	PPD+
MovieLens	NDCG@1	0.00348	0.01142	**0.02666**
	NDCG@5	0.00904	0.03239	**0.06449**
	NDCG@10	0.01270	0.04732	**0.08896**
	HR@5	0.01490	0.05414	**0.10331**
	HR@10	0.02632	0.10083	**0.17947**
	MRR	0.01548	0.04503	**0.08134**
Goodreads	NDCG@1	**0.00416**	0.00337	**0.00416**
	NDCG@5	0.00995	0.00922	**0.01090**
	NDCG@10	0.01315	0.01299	**0.01529**
	HR@5	0.01574	0.01503	**0.01749**
	HR@10	0.02579	0.02682	**0.03129**
	MRR	0.01263	0.01339	**0.01493**
UserBehavior	NDCG@1	0.03935	0.04320	**0.04374**
	NDCG@5	0.06221	0.07018	**0.07118**
	NDCG@10	0.07018	0.07944	**0.08050**
	HR@5	0.08339	0.09503	**0.09654**
	HR@10	0.10816	0.12373	**0.12549**
	MRR	0.06392	**0.07172**	0.07167

In addition to the baselines, PPD+ can retrieve past interactions by comparing groups of items as the sub-sequences retrieved from leveraging detected drift points. The model exploits the Principal Component Analysis to compute the representation for comparing between sub-sequences in terms of items' distribution. The model benefits from extracting the user preference concerning items' distribution since it interprets an overview of the user preference. It further allows the model to capture relations of items within the same subsequence. In contrast, the baselines retrieve past interactions by comparing the latest item with other items from the same group. As a result, the baselines' prediction will be highly influenced by only a single item and may not reflect the actual user preference. For example, the user may be interested in healthy food, and the latest action is a salad. The baselines may suggest items related to vegetables instead of nutritious food.

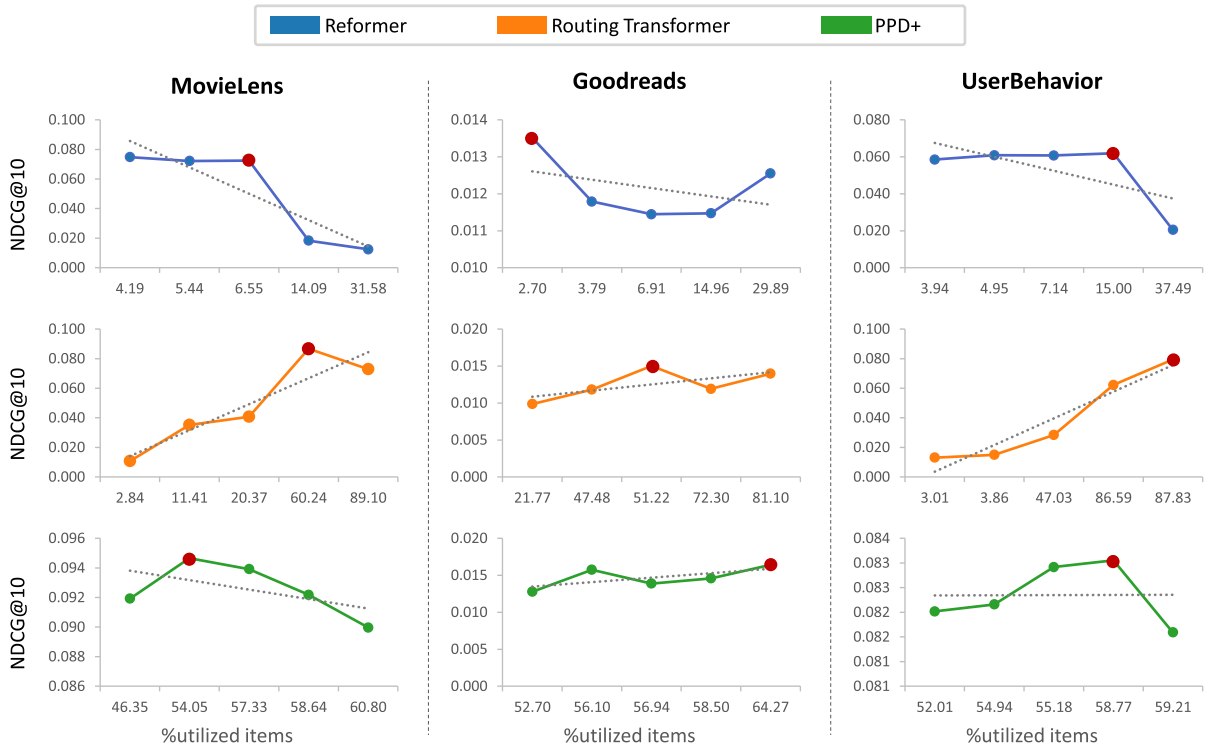


FIGURE 9. The impact of the percentage of utilized items in the historical sequence on the model's performance. The red point indicates the point with the highest NDCG@10 value, and the gray line is the trend line of NDCG@10.

3) IMPACT OF NOISES

Since user preferences are dynamic and change over time, some past interactions may not be relevant to the current user preference. Considering interactions irrelevant to the current user preference will introduce noises in the model's inputs and result in irrelevant recommendations. To study the impact of noise reduction on the model's input with the performance, we illustrate the evaluation results of CBT and PPD+ in terms of NDCG@10 (y-axis) with the average percentage of utilized items (x-axis) in Fig. 9. The red point in Fig. 9 represents the point with the highest NDCG@10. The gray line is the trend line between the NDCG@10 and the average percentage of utilized items.

In Routing Transformer, the trend line in every dataset is positive, showing that a higher percentage of utilized items leads to higher NDCG@10. In contrast, the trend of NDCG@10 in Reformer decreases when the average percentage of utilized items increases. A similar trend also occurs with PPD+, where it shows a negative trend in MovieLens, a steady trend in UserBehavior, and a positive trend in Goodreads.

Although a higher percentage of utilized items provides more information to the model, most of the highest NDCG@10 is not the point with the highest percentage of utilized items. The poor performance when utilizing more items demonstrates the drawbacks of noises in the historical sequence. We further notice that the NDCG@10 of Reformer and PPD+ for MovieLens and UserBehavior are worst when the average percentage of utilized items is the highest. Therefore, every model has the optimal value of the average percentage of utilized items for delivering relevant recommendations. The surplus information may increase noises as the unrelated information to the model's input and result in irrelevant recommendations.

B. CASE STUDIES

To demonstrate the model's mechanism when applied in different scenarios, we present four example cases and discuss the behavior of each model under these circumstances. The first case is when the latest interaction is entirely irrelevant to every past interaction in the historical sequence. In the second case, we discuss the scenario where the user preference is incrementally changed. Next, the third case demonstrates the impact on recommendations when the model incorrectly grouping similar items. Lastly, the fourth case focuses on the model's behavior when grouping similar items by hard clustering and soft clustering. In the fourth case, we will discuss only Routing Transformer, Reformer, and PPD+ since other models do not leverage the clustering algorithm.

In every case, the example historical sequence is based on five movies (m_1 to m_5) with different genres. The genres of movies in every case are action, fantasy, drama, and horror. Some movies may contain several genres to illustrate the situation in a real-world scenario. For clarity, we also show the item utilization of each model, where a utilized item is the item with the correct mark (✓). Otherwise, an item without a correct mark is ignored by a particular model. Furthermore, we present both variations of Longformer in

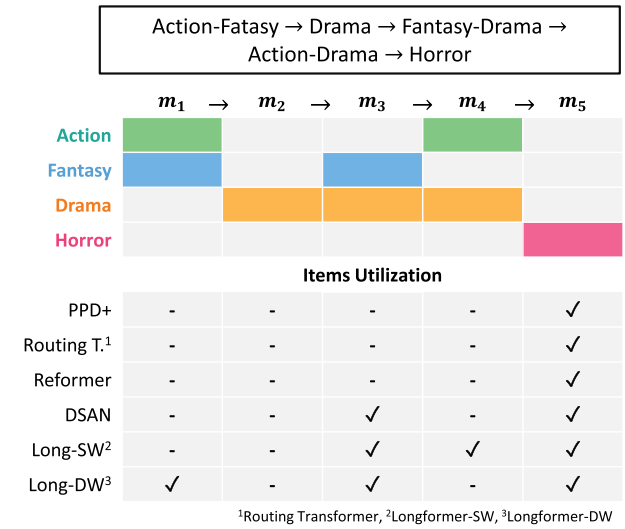


FIGURE 10. Example of a case where the latest interaction is completely irrelevant to other past interactions.

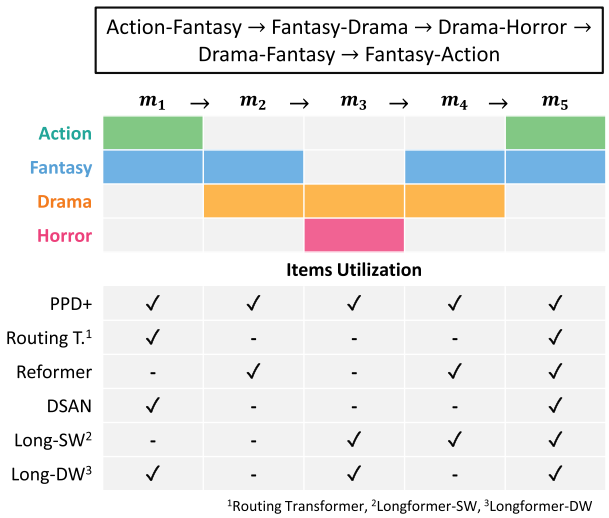


FIGURE 11. Example of a case where the user preference is incrementally changed.

every case, which are Longformer-SW and Longformer-DW. We set the length of a sliding window of Longformer-SW as three to represent the utilization of around half of the whole historical sequence. Similarly, Longformer-DW has a dilated size of one to simulate the case where it considers only half of the whole historical sequence.

1) IRRELEVANT LATEST INTERACTION

Fig. 10 illustrates a scenario where the latest interaction is completely irrelevant to every past interaction. The latest interaction (m_5) is a movie from the horror genre, whereas other interactions are action, fantasy, and drama movies. In this case, PPD+ will indicate that the drift point occurred at the latest interaction. PPD+ will ignore other past interactions since none of the past interactions are similar to the latest interaction. Consequently, PPD+ will deliver the recommendation based on the probability of the next movie after watching the latest movie (m_5). Similarly, Routing Transformer and Reformer will have the same behavior as PPD+, where they deliver the recommendation by ignoring other past interactions and consider only the latest interaction.

In contrast, DSAN will consider some past interactions due to its sparse activation function, which enforces some past interactions to have a high attention value. However, none of the past interactions are relevant to the current user preference. In this case, the movie with high attention score is not a movie that is similar to the latest movie. As shown in Fig. 10, the utilized movie is m_3 which is a fantasy-drama movie, whereas the latest movie is a horror movie. Hence, a fantasy-drama movie will be noise in the input of DSAN and result in a recommendation of a movie with irrelevant genres to current user preference.

For Longformer, most movies are utilized since the attention pattern of Longformer is pre-defined based on the position of interaction in the historical sequence. The pre-defined pattern forced the model to consider movies from irrelevant genres. Specifically, the attention pattern of Longformer-SW is based on sliding windows. Hence, Longformer-SW will consider m_3, m_4, and m_5, which include irrelevant information related to action, fantasy, and drama movies. In the same way, Longformer-DW will consider m_1, m_3, and m_5 due to the dilated size of one, which results in noisy input from action, fantasy, and drama movies.

2) INCREMENTALLY CHANGED USER PREFERENCE

The scenario where user preference is incrementally changed is depicted in Fig. 11. There is a mutual genre for every consecutive movie, showing that the user preference is slowly changed from one genre to another. For instance, both m_1 and m_2 belong to the fantasy genre. Since PPD+ leverages soft clustering to detect the significant changes in consecutive movies' characteristics, the latent classes of consecutive movies will not be different. The difference between characteristics of consecutive movies will be small since they have mutual characteristics as a genre. Consequently, PPD+ may not detect any signal of user preference drift, and the model will utilize every past interaction to deliver the recommendation.

In the case of Routing Transformer and Reformer, the drift point may occur at every interaction due to the change of the movie's group since the movie's genres are changed. Notice that Routing Transformer and Reformer consider different movies for the recommendation since both models may categorize the same item into different groups. On the one hand, Routing Transformer may categorize m_1 and m_5 into the same group base on the action genre. On the other hand, Reformer may categorize m_2, m_4, and m_5 into the same group based on their mutual fantasy genre. However, Routing Transformer and Reformer will retrieve past interactions as a single movie rather than a period of various movies. When retrieving a movie as a single item, the model will ignore the relation between items and may not capture some aspects of the user

preference. For instance, Routing Transformer will ignore the transition relation among movies in the genre of action-fantasy ($m_1 \rightarrow m_2$), and fantasy-drama ($m_2 \rightarrow m_3$). Moreover, the recommendations from Routing Transformer, when ignoring m_2, m_3, and m_4, will not include user preferences related to the drama genre.

For DSAN, the model will utilize highly similar movies and ignore other movies. As shown in Fig. 11, the model may eventually select only the first movies since it has a similar genre as the latest interaction. Nonetheless, there are no significantly similar movies since every movie shares a mutual genre with its previous movie. Therefore, DSAN will ignore most aspects of user preference and result in irrelevant recommendations. For instance, DSAN will ignore a preference which is a mixture between fantasy and drama movies, which are contained in m_2 and m_4.

In contrast, Longformer will consider most of the past interactions due to the fixed attention pattern. As shown in Fig. 11, Longformer-SW with a sliding window will consider m_3, m_4, and m_5. Whereas, Longformer-DW with a dilated size of one will consider m_1, m_3, and m_5. However, it may capture only some user preferences while ignoring other useful aspects of the user preference. For example, Longformer-DW skips m_2 and m_4 such that the preference related to fantasy-drama movies is completely ignored.

3) INCORRECTLY GROUPING SIMILAR ITEMS

Fig. 12 illustrates the impact of incorrectly grouping similar items. In this scenario, the last three movies share a mutual genre and can be viewed as movies from a similar group. Specifically, m_3 and m_4 share a common drama genre, while both m_4 and m_5 belong to the fantasy genre. Moreover, m_3 and m_5 have a mutual characteristic of action movies. Therefore, these three movies have similar characteristics, and their representation should be near to each other in the latent space.

However, these three items may not belong to the same group in the case of hard clustering as in Routing Transformer and Reformer. As shown in Fig. 12, m_3 and m_4 may be categorized into the same group, whereas m_5 is in another group with m_1. Consequently, Routing Transformer and Reformer will utilize only m_1, which ignore the current user preference related to drama movie. The recommendation from these two models will be only action-fantasy movies rather than action-fantasy-drama movies.

Similarly, DSAN also encounters the same issue as Routing Transformer and Reformer due to its sparse attention pattern. Since DSAN considers only movies with high attention scores with the latest movies, it will utilize only m_1 in this scenario. Therefore, DSAN will not have a movie with a drama genre in its recommendations.

In contrast, PPD+ with soft clustering will group these three items into the same sub-sequence since their characteristics are not significantly different. As a result, PPD+ will consider m_1, m_3, m_4, and m_5 to deliver the recommendation, which is the movie includes preference from action, fantasy, and drama.

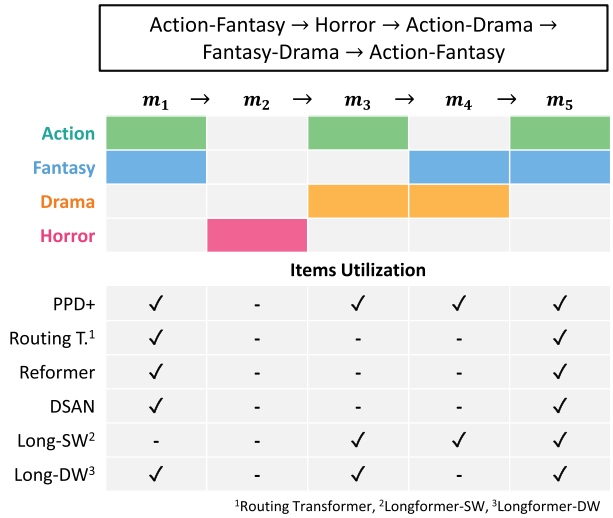


FIGURE 12. Example of a case where the grouping of items is incorrect.

In the case of Longformer, although the attention pattern may group m_3, m_4, and m_5 together as in Longformer-SW, the model is not correctly utilizing past interactions. Specifically, Longformer-SW ignores m_1, which is also similar to the latest interaction. Additionally, Longformer-DW regardless of m_4, which contains user preferences about the fantasy genre and further provides information about user preferences related to fantasy-drama movies.

4) IMPACT OF DIFFERENT CLUSTERING ALGORITHMS

In this case study, we will focus only on the model which utilizes a clustering algorithm, i.e., Routing Transformer, Reformer, and PPD+. Fig. 13 depicts the example scenario where we assume that the number of clusters (NC) is four according to the number of genres. The groups of each movie are indicated by color in the lower left part of Fig. 13. Additionally, the illustration of the decision boundary for grouping movies with similar characteristics is shown on the right-hand side of Fig. 13.

In PPD+, the groups of movies can be a mixture of different groups since it leverages the soft clustering, which allows movies to exist in various groups simultaneously. PPD+ can naturally capture the movie's characteristics in a real-world scenario, where a movie can have several genres. For instance, m_1 has genres of action, fantasy, and horror. The groups of m_1 are a mixture of three genres, indicated by the gradient of three colors (green, blue, and pink). Moreover, the position of m_1 in the latent space is the intersection area between three groups, showing that it belongs to all three groups.

The correct capturing of movies' characteristics benefits PPD+ to model user preference accurately. Given the scenario in Fig. 13, the latest interaction is the movie with action and fantasy genres. PPD+ will utilize m_1, m_3, m_4, and m_5 since they are near each other in the latent space, as shown in Fig. 13. Consequently, the recommendation from PPD+

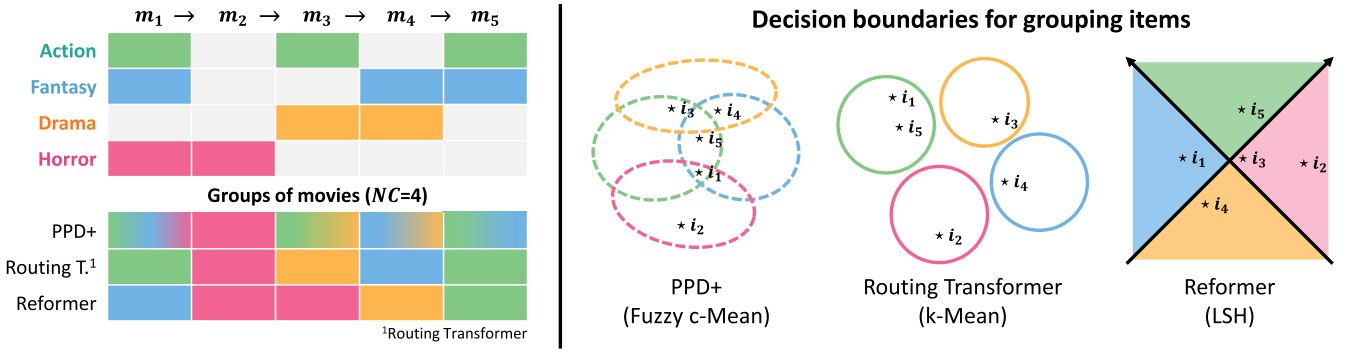


FIGURE 13. An illustration of the clustering algorithm in PPD+, Routing Transformer, and Reformer. The left-hand side of the figure describes groups of movies, while the right-hand side shows each model's decision boundaries.

will be a movie having multiple genres which cover various aspects of user preference, such as action, fantasy, and drama.

In contrast, Routing Transformer and Reformer utilize a hard clustering algorithm as k-Mean and LSH. Consequently, movies are forced to belong to only one group, which does not reflect the actual characteristic of movies. As the decision boundaries are shown in Fig. 13, Routing Transformer and Reformer categorize each movie into a single group. Therefore, the hard clustering is not correctly captured the movie's characteristics in a real-world scenario.

Moreover, Routing Transformer and Reformer allow only movies from the same group to establish relations with each other. These models are limited to integrating preferences of similar movies from different groups as in PPD+. Hence, the recommendations from Routing Transformer and Reformer ignore preferences that are related to current user preference. For instance, Routing Transformer will focus only on m_1 since it was categorized into the same group as m_5. At the same time, Reformer will consider only the latest interaction and ignore other past interactions.

It is also worth mentioning the flexibility of PPD+ and Routing Transformer to optimize centroids for clustering along with the recommendation task. These two models can optimize centroids to represent groups of similar movies in a real-world scenario. On the other hand, centroids in Reformer cannot be optimized, which limits Reformer from modeling the correct movie representation. As shown in Fig. 13, m_3 is categorized into the horror group (pink), although it has action and drama genres. The incorrect categorization of movies is due to bad decision boundaries from fixed centroids of Reformer. Specifically, m_3 should be in the middle of the action group (green) and drama group (yellow) since it has both of these genres. However, the only available groups between action and drama groups are fantasy and horror groups, which both are the incorrect group. The bad centroids forced m_3 to be in the horror group, although it does not have characteristics of a horror movie. Consequently, the Reformer may not capture correct user preference since the movie representation does not express its actual characteristics and eventually results in irrelevant recommendations.

C. IMPACT OF THRESHOLD FOR OPTIMAL ITEM UTILIZATION

In this work, the proposed method selects the optimal number of utilized items based on the threshold and the cosine similarity between other sub-sequences and the latest sub-sequence. To evaluate the effectiveness of the proposed threshold for optimal item utilization, we performed an additional experiment and created two variants of PPD+ as follows:

- **M-PPD+**: To retrieve sub-sequences with preferences more relevant than the average of every sub-sequence, this model computes the average (avg_{sim}) of the results from (40) and utilizes it as the threshold. The model retrieves every sub-sequence having a cosine similarity with the latest sub-sequence more than the avg_{sim}.
- **MS-PPD+**: This model filters to retrieve only sub-sequences with preferences significantly more relevant than the average of every sub-sequence. It computes the threshold as the summation of the average (avg_{sim}) and the standard deviation (sd_{sim}) of the result from (40). Only sub-sequences with the cosine similarity between the latest sub-sequence higher than the $avg_{sim} + sd_{sim}$ are selected as the relevant sub-sequence.

To focus only on the impact of the threshold, we utilized the same hyper-parameters for every model. We conducted the experiment by following the same settings mentioned in Section IV. We report the evaluation results on the validation set of every model in Table 5.

According to the results in Table 5, M-PPD+ outperforms MS-PPD+ in most metrics across every dataset. Besides, PPD+ has surpassed evaluation results than the variants in most metrics across every dataset. The superior results of PPD+ show that the threshold of PPD+ is more effective than the variant models. The threshold of PPD+ yields the past interactions with the cosine similarity higher than zero, which ensures that the selected sub-sequences are similar to the latest sub-sequence.

On the contrary, although the threshold of M-PPD+ enforces to retrieve more significant similar sub-sequences, M-PPD+ may select sub-sequences dissimilar to the latest sub-sequence. Since the cosine similarity value is

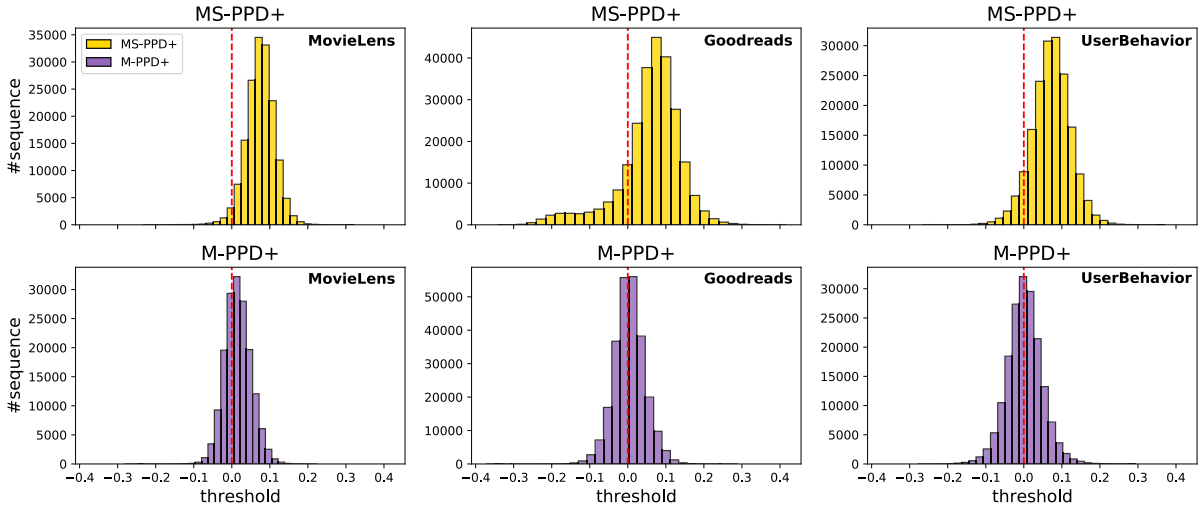


FIGURE 14. The illustration of threshold distributions in MS-PPD+ and M-PPD+. The red vertical line represents the point where the threshold is zero.

TABLE 5. Comparison of the evaluation results from variants of the proposed method.

	Metrics	MS-PPD+	M-PPD+	PPD+
MovieLens	NDCG@1	0.0273/	0.03079	**0.03162**
	NDCG@5	0.06883	0.07067	**0.07121**
	NDCG@10	0.09169	0.09172	**0.09466**
	HR@5	0.11076	0.11126	**0.11192**
	HR@10	0.18195	0.17666	**0.18510**
	MRR	**0.08594**	0.08167	0.08476
Goodreads	NDCG@1	0.00349	0.00337	**0.00363**
	NDCG@5	0.00913	0.00972	**0.01099**
	NDCG@10	0.01303	0.01356	**0.01559**
	HR@5	0.01497	0.01613	**0.01833**
	HR@10	0.02715	0.02812	**0.03278**
	MRR	0.01317	0.01324	**0.01557**
UserBehavior	NDCG@1	0.04469	0.04525	**0.04656**
	NDCG@5	0.07129	0.07328	**0.07390**
	NDCG@10	0.08045	0.08279	**0.08301**
	HR@5	0.09616	0.09919	**0.09942**
	HR@10	0.12459	**0.12875**	0.12768
	MRR	0.07257	0.07452	**0.07486**

between −1 and 1, the average of the cosine similarity can be less than zero. Consequently, M-PPD+ may select dissimilar sub-sequences, which contain a preference irrelevant to the current user preference. We illustrate the threshold distributions of each model in Fig. 14, where the red vertical line represents the point with the threshold equal to zero. According to Fig. 14, the threshold distributions of M-PPD+ are located around zero, while around half of the distributions are negative values which cause the model to retrieve dissimilar sub-sequences.

In MS-PPD+, the threshold distributions are slightly shifted toward a positive value, as shown in Fig. 14. The threshold has a higher chance of being a positive number since the threshold includes the standard deviation, which is certainly a positive number. Nonetheless, the number of

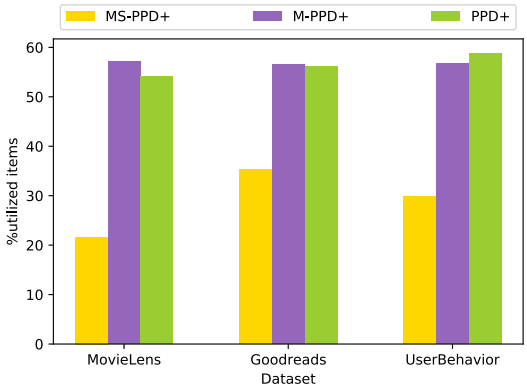


FIGURE 15. The percentage of utilized items in MS-PPD+, M-PPD+, and PPD+ across every dataset.

selected sub-sequences may be insufficient due to the high threshold, causing the model to lack the past information for delivering the recommendations. On the other hand, PPD+ is not suffering from insufficient past information as in MS-PPD+ since the threshold is optimal and guarantees to provide sub-sequences containing preference relevant to the current user preference. As shown in Fig. 15, the percentages of utilized items in MS-PPD+ are only around 30 percent, which may not be sufficient for delivering the prediction. In contrast, M-PPD+ and PPD+ use about 50 percent of the total interactions as an optimal percentage of utilized items.

VI. CONCLUSION

We proposed a unified sequential recommender system called PPD+. The proposed method detects the personalized drift pattern of the user preference by grouping similar items with soft label and utilizes the optimal number of relevant interactions. PPD+ overcomes the requirement of a hyper-parameter for selecting past interactions by considering the optimal

amount of relevant interactions based on similarities with the current user preference. To correspond with user preference dynamics, PPD+ determines personalized drift patterns by comparing item characteristics as soft label classes from the Fuzzy c-Mean algorithm. The drift points are leveraged to divide the historical sequence into many sub-sequences, where each sub-sequence contains only a particular user preference. To deliver the recommendation relevant to the current user preference, PPD+ considers only sub-sequences containing user preferences similar to the latest sub-sequence. PPD+ compares sub-sequences with an unequal number of items by employing the Principal Component Analysis and cosine similarity. PPD+ retrieves sub-sequences with cosine similarity greater than zero to ensure that the selected sub-sequences are similar to the current user preference. Furthermore, PPD+ is trained in an end-to-end approach by combining the loss function of clustering and prediction as a unified objective.

The experiments show that PPD+ is more effective in grouping similar past interactions and delivers more relevant recommendations than the baselines. We further performed an additional experiment by modifying Content-Based Transformers to consider an equal number of items to remove the impact of utilizing a different amount of items. The results suggest the superior of PPD+ on grouping items with soft clustering and benefits from comparing groups of items in terms of distribution.

To verify the competence of the threshold value for optimal item utilization, we experimented with the variants of PPD+. Each variant model has different personalized values of the threshold. According to the results, PPD+ using the threshold as zero shows superior results than utilizing the mean and the mean with standard deviation since the threshold of PPD+ ensures to retrieve only similar past interactions.

In future work, we are interested in leveraging the information from irrelevant interactions. Since irrelevant interactions may contain latent user preferences as the user was interested and interacted with them in the past. We aim to discover an effective method to extract only helpful user preferences from irrelevant interactions while excluding noise information.

REFERENCES

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

[2] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in Neural Information Processing Systems*, vol. 20, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. Red Hook, NY, USA: Curran Associates, 2007.

[3] D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems*, vol. 13, T. Leen, T. Dietterich, and V. Tresp, Eds. Cambridge, MA, USA: MIT Press, 2000.

[4] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web (WWW)*, Geneva, Switzerland, 2017, pp. 173–182.

[5] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, "XDeepFM: Combining explicit and implicit feature interactions for recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 1754–1763.

[6] M. Khoali, A. Tali, and Y. Laaziz, "Advanced recommendation systems through deep learning," in *Proc. 3rd Int. Conf. Netw., Inf. Syst. Secur. (NISS)*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 1–8.

[7] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec: Autoencoders meet collaborative filtering," in *Proc. 24th Int. Conf. World Wide Web (WWW)*. New York, NY, USA: Association for Computing Machinery, 2015, pp. 111–112.

[8] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-n recommender systems," in *Proc. 9th ACM Int. Conf. Web Search Data Mining (WSDM)*. New York, NY, USA: Association for Computing Machinery, 2016, pp. 153–162.

[9] F. Yang and Y. Lu, "Restricted Boltzmann machines for recommender systems with implicit feedback," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 4109–4113.

[10] C. Chen, M. Zhang, Y. Liu, and S. Ma, "Neural attentional rating regression with review-level explanations," in *Proc. World Wide Web Conf. (WWW)*, Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2018, pp. 1583–1592.

[11] C. Panagiotakis, H. Papadakis, A. Papagrigoriou, and P. Fragopoulou, "Improving recommender systems via a dual training error based correction approach," *Expert Syst. Appl.*, vol. 183, Nov. 2021, Art. no. 115386.

[12] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, "Recurrent recommender networks," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, Feb. 2017, pp. 495–503.

[13] T. Donkers, B. Loepp, and J. Ziegler, "Sequential user-based recurrent neural network recommendations," in *Proc. 11th ACM Conf. Rec. Syst. (RecSys)*. New York, NY, USA: Association for Computing Machinery, 2017, pp. 152–160.

[14] B. Hidasi and A. Karatzoglou, "Recurrent neural networks with top-k gains for session-based recommendations," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2018, pp. 843–852.

[15] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, "Personalizing session-based recommendations with hierarchical recurrent neural networks," in *Proc. 11th ACM Conf. Rec. Syst. (RecSys)*. New York, NY, USA: Association for Computing Machinery, 2017, pp. 130–137.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst. (NIPS)*. Red Hook, NY, USA: Curran Associates, 2017, pp. 6000–6010.

[17] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2018, pp. 197–206.

[18] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1441–1450.

[19] J. Yuan, Z. Song, M. Sun, X. Wang, and W. X. Zhao, "Dual sparse attention network for session-based recommendation," in *Proc. AAAI Conf. Artif. Intell. CIKM*, May 2021, vol. 35, no. 5, pp. 4635–4643.

[20] J. Basilico and Y. Raimond, "Déjà vu: The importance of time and causality in recommender systems," in *Proc. 11th ACM Conf. Rec. Syst. (RecSys)*. New York, NY, USA: Association for Computing Machinery, 2017, p. 342.

[21] J. Zhao, P. Zhao, L. Zhao, Y. Liu, V. S. Sheng, and X. Zhou, "Variational self-attention network for sequential recommendation," in *Proc. IEEE 37th Int. Conf. Data Eng. (ICDE)*, Los Alamitos, CA, USA, Apr. 2021, pp. 1559–1570.

[22] B. Peters, V. Niculae, and A. F. T. Martins, "Sparse sequence-to-sequence models," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 1504–1519.

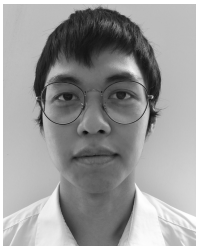
[23] D. Neil, M. Pfeiffer, and S.-C. Liu, "Phased LSTM: Accelerating recurrent network training for long or event-based sequences," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst. (NIPS)*. Red Hook, NY, USA: Curran Associates, 2016, pp. 3889–3897.

[24] Y. Zhu, H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, and D. Cai, "What to do next: Modeling user behaviors by time-LSTM," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, Aug. 2017, pp. 3602–3608.

[25] N. Sritrakool and S. Maneeroj, "Personalized preference drift aware sequential recommender system," *IEEE Access*, vol. 9, pp. 155491–155506, 2021.

[26] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16×16 words: Transformers for image recognition at scale," in *Proc. ICLR*, 2021, pp. 1–21. [Online]. Available: https://dblp.org/rec/conf/iclr/DosovitskiyB0WZ21.html?view=bibtex
[27] L. Dong, S. Xu, and B. Xu, "Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 5884–5888.
[28] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," 2020, *arXiv:2004.05150*.
[29] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–11. [Online]. Available: https://dblp.org/rec/conf/iclr/KitaevKL20.html?view=bibtex&param=1
[30] A. Roy, M. Saffar, A. Vaswani, and D. Grangier, "Efficient content-based sparse attention with routing transformers," *Trans. Assoc. Comput. Linguistics*, vol. 9, pp. 53–68, Feb. 2021.
[31] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[32] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734.
[33] Y. Han, Q. Li, Y. Xiao, H. Zhou, Z. Yang, and J. Wu, "Multiple interleaving interests modeling of sequential user behaviors in e-commerce platform," *World Wide Web*, vol. 24, no. 4, pp. 1121–1146, Jul. 2021.
[34] C. Yan, Y. Wang, Y. Zhang, Z. Wang, and P. Wang, "Modeling Long- and short-term user behaviors for sequential recommendation with deep neural networks," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2021, pp. 1–8.
[35] J. Zhang, C. Ma, C. Zhong, X. Mu, and L. Wang, "MBPI: Mixed behaviors and preference interaction for session-based recommendation," *Appl. Intell.*, vol. 51, no. 10, pp. 7440–7452, Oct. 2021.
[36] H. Wang, K. Yao, J. Luo, and Y. Lin, "An implicit preference-aware sequential recommendation method based on knowledge graph," *Wireless Commun. Mobile Comput.*, vol. 2021, pp. 1–12, Aug. 2021.
[37] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," in *Proc. 11th ACM Int. Conf. Web Search Data Mining (WSDM)*. New York, NY, USA: Association for Computing Machinery, 2018, pp. 565–573.
[38] S. Yakhchi, A. Behehsti, S.-M. Ghafari, I. Razzak, M. Orgun, and M. Elahi, "A convolutional attention network for unifying general and sequential recommenders," *Inf. Process. Manage.*, vol. 59, no. 1, Jan. 2022, Art. no. 102755.
[39] Q. Zhang, L. Cao, C. Shi, and Z. Niu, "Neural time-aware sequential recommendation by jointly modeling preference dynamics and explicit feature couplings," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Apr. 14, 2021, doi: 10.1109/TNNLS.2021.3069058.
[40] W. Chen and H. Chen, "Collaborative co-attention network for session-based recommendation," *Mathematics*, vol. 9, no. 12, p. 1392, Jun. 2021.
[41] C.-H. Lee, J.-E. Ding, C.-M. Chen, J.-K. Lou, M.-F. Tsai, and C.-J. Wang, "LSTPR: Graph-based matrix factorization with long short-term preference ranking," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.* New York, NY, USA: Association for Computing Machinery, 2021, pp. 2222–2226.
[42] D. Hu, L. Wei, W. Zhou, X. Huai, Z. Fang, and S. Hu, "PEN4Rec: Preference evolution networks for session-based recommendation," in *Knowledge Science, Engineering and Management*, H. Qiu, C. Zhang, Z. Fei, M. Qiu, and S.-Y. Kung, Eds. Cham, Switzerland: Springer, 2021, pp. 504–516.
[43] W. Chen, P. Ren, F. Cai, F. Sun, and M. De Rijke, "Multi-interest diversification for end-to-end sequential recommendation," *ACM Trans. Inf. Syst.*, vol. 40, no. 1, pp. 1–30, Jan. 2022.
[44] E. Shao, S. Guo, and Z. A. Pardos, "Degree planning with plan-bert: Multi-semester recommendation using future courses of interest," in *Proc. AAAI Conf. Artif. Intell.*, May 2021, vol. 35, no. 17, pp. 14920–14929.
[45] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, and A. Ahmed, "Big bird: Transformers for longer sequences," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 17283–17297. [Online]. Available: https://papers.nips.cc/paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html
[46] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlós, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, D. Belanger, L. J. Colwell, and A. Weller, "Rethinking attention with performers," 2020, *arXiv:2009.14794*.
[47] C. Wu, F. Wu, T. Qi, Y. Huang, and X. Xie, "Fastformer: Additive attention can be all you need," 2021, *arXiv:2108.09084*.
[48] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are RNNs: Fast autoregressive transformers with linear attention," in *Proc. 37th Int. Conf. Mach. Learn.*, in Proceedings of Machine Learning Research, vol. 119, H. Daume, III, and A. Singh, Eds., Jul. 2020, pp. 5156–5165.
[49] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," 2019, *arXiv:1904.10509*.
[50] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: A survey," 2020, *arXiv:2009.06732*.
[51] L. J. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," Jul. 2016, *arXiv:1607.06450*.
[52] M. Wan and J. J. McAuley, "Item recommendation on monotonic behavior chains," in *Proc. 12th ACM Conf. Rec. Syst. (RecSys)*, Vancouver, BC, Canada, Oct. 2018, pp. 86–94.
[53] M. Wan, R. Misra, N. Nakashole, and J. J. McAuley, "Fine-grained spoiler detection from large-scale review corpora," in *Proc. 57th Conf. Assoc. Comput. Linguistics (ACL)*, vol. 1, Florence, Italy: Association for Computational Linguistics, Jul./Aug. 2019, pp. 2605–2610.
[54] H. Zhu, D. Chang, Z. Xu, P. Zhang, X. Li, J. He, H. Li, J. Xu, and K. Gai, "Joint optimization of tree-based index and deep model for recommender systems," in *Advances in Neural Information Processing Systems*, vol. 32. Red Hook, NY, USA: Curran Associates, 2019.
[55] H. Zhu, X. Li, P. Zhang, G. Li, J. He, H. Li, and K. Gai, "Learning tree-based deep model for recommender systems," *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 1079–1088. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3219819.3219826, doi: 10.1145/3219819.3219826.
[56] J. Zhuo, Z. Xu, W. Dai, H. Zhu, H. Li, J. Xu, and K. Gai, "Learning optimal tree models under beam search," in *Proc. ICML*, 2020, pp. 11650–11659. [Online]. Available: https://dl.acm.org/doi/10.5555/3524938.3526018, doi: 10.5555/3524938.3526018.
[57] J. Shen *et al.*, "Lingvo: A modular and scalable framework for sequence-to-sequence modeling," 2019, *arXiv:1902.08295*.
[58] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, Dec. 2014, pp. 1–15. [Online]. Available: https://dblp.org/rec/journals/corr/KingmaB14.html

SARANYA MANEEROJ received the B.S. degree from Chulalongkorn University, Thailand, in 1996, and the M.E. and Dr. (Eng.) degrees from The University of Electro-Communications, Japan, in 2001 and 2005, respectively. She is currently an Associate Professor with the Department of Mathematics and Computer Science, Faculty of Science, Chulalongkorn University. Her research interests include recommender systems and data mining.



NAKARIN SRITRAKOOL is currently pursuing the bachelor's degree in computer science with Chulalongkorn University, Thailand. In 2021, he did an internship at Sertis Company Ltd. His research interests include recommender systems and machine learning.