Directional and Explainable Serendipity Recommendation

Xueqi Li
lee_xq@hnu.edu.cn
Hunan University

Wenjun Jiang*
jiangwenjun@hnu.edu.cn
Hunan University

Weiguang Chen
cwg@hnu.edu.cn
Hunan University

Jie Wu
jiewu@temple.edu
Temple University

Guojun Wang
csgjwang@gzhu.edu.cn
Guangzhou University

Kenli Li
lkl@hnu.edu.cn
Hunan University

ABSTRACT

Serendipity recommendation has attracted more and more attention in recent years; it is committed to providing recommendations which could not only cater to users' demands but also broaden their horizons. However, existing approaches usually measure user-item relevance with a scalar instead of a vector, ignoring *user preference direction*, which increases the risk of unrelated recommendations. In addition, reasonable explanations increase users' trust and acceptance, but there is no work to provide explanations for serendipitous recommendations. To address these limitations, we propose a Directional and Explainable Serendipity Recommendation method named **DESR**. Specifically, we extract users' long-term preferences with an unsupervised method based on GMM (Gaussian Mixture Model) and capture their short-term demands with the capsule network at first. Then, we propose the *serendipity vector* to combine long-term preferences with short-term demands and generate directionally serendipitous recommendations with it. Finally, a back-routing scheme is exploited to offer explanations. Extensive experiments on real-world datasets show that **DESR** could effectively improve the serendipity and explainability, and give impetus to the diversity, compared with existing serendipity-based methods.

KEYWORDS

Recommendation, Serendipity, User Preference Direction, Explainability

ACM Reference Format:
Xueqi Li, Wenjun Jiang, Weiguang Chen, Jie Wu, Guojun Wang, and Kenli Li. 2020. Directional and Explainable Serendipity Recommendation. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380100

1 INTRODUCTION

With the extensive application of deep learning in recommender systems [7, 8, 11, 39], there has been an unprecedented improvement on recommendation accuracy, exacerbating over-specialization. To deal with this issue, researchers propose some serendipity-oriented methods [3, 9, 26, 31] which are expected to recommend interesting items beyond users' discovery. So far, it is still challenging to find serendipitous items due to the collision between difference and accuracy (both are pursued by serendipity recommendation) [21].

To address the challenge, some existing methods propose to gather some basic results [23, 30, 42] such as novel, unexpected, and valuable items [23] as serendipitous recommendations. These items may only meet either the accuracy or difference requirements, hardly being regarded as serendipitous ones. To deal with this problem, some others [24, 26] generate serendipitous items based on the level of user-item relevance: the items with a stronger association to users are accurate recommendations, the items without associations are unrelated items, and there are serendipitous items between the two, which could appeal to users but are beyond their discovery. However, these methods measure user-item relevance with a scalar, ignoring the guiding role of *user preference direction*. To recommend more serendipitous items and increase user acceptance, we propose a Directional and Explainable Serendipity Recommendation method (**DESR**).

The following is an example to illustrate the effect that our approach tries to reach, based on a randomly selected user on MovieLens-1m[1] [12] (UserID: 4824). As is shown in Fig. 1(a), for the target user u_{tar}, there are five preference directions (we set the direction from short-term demand to long-term preference as a preference direction). Accuracy-oriented methods (**ACC**) tend to recommend those items that are very close to u_{tar}'s short-term demand, (i.e., t_5, t_6). **HAES** [24], a recent serendipity recommendation method, would be apt to recommend items (i.e., t_0, t_1, t_2). It focuses on determining a suitable distance between serendipitous recommendations and the short-term demand (i.e., the radius of the ring), but ignores *user preference direction* (i.e., the directions of arrows). However, users are only interested in those items with particular characteristics. As is illustrated in Fig. 1(c), there are significantly distinct clusters of user-related items. Considering users' preferences in the figure, t_1 and t_2 would be possibly unrelated recommendations. To deal with this problem, we attempt to integrate the demand-recommendation distance and *user preference direction* as a whole named *serendipity vector* (i.e., the arrows in Fig. 1(b)). **DESR** would allocate a recommendation area for each *serendipity vector*, of which the ones covering the items in Fig. 1(b) are drawn by ellipses. t_0 and t_4 are the optimal recommendations.

As a key element of **DESR**, users' preferences are usually extracted by neural methods, such as Recurrent Neural Network [5, 10, 29] and Graph Neural Network [6, 16, 38], laying a solid

*Wenjun Jiang is the corresponding author.

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380100

[1] https://grouplens.org/datasets/movielens/1m/

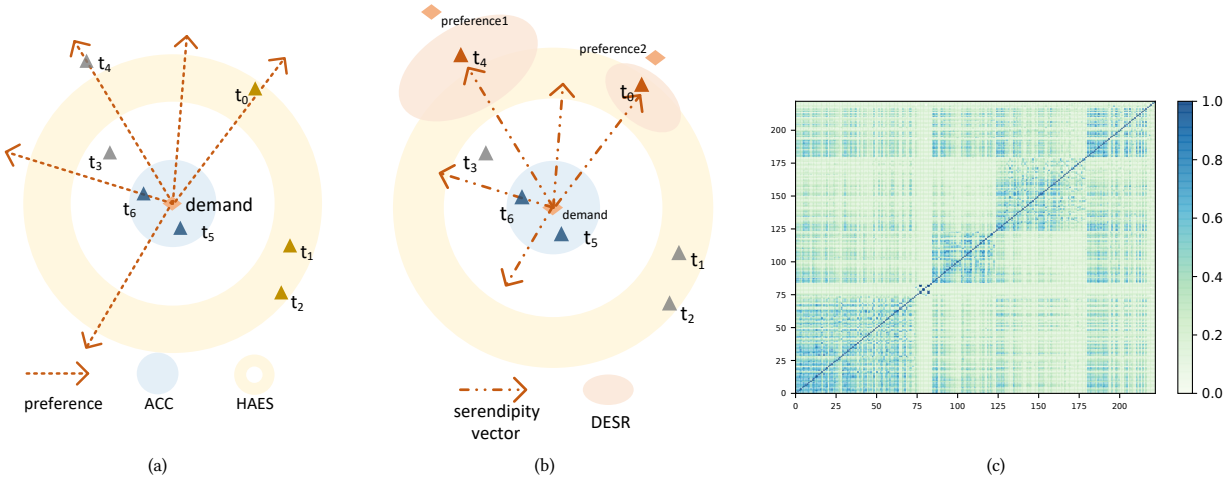


Figure 1: An example to illustrate the effect our approach tries to reach.

foundation for accuracy recommendation. They try to make the extracted preference and the ground truth preference label as similar as possible in the process of training. Most neural methods usually optimize for a specific task; these methods have a poor performance on guaranteeing the generality and integrity of users' preferences. For example, if a direction (from *demand* to $preference_0$ in Fig. 1(a)) is set as labeled preference direction in the training of RNN, the more similar the labeled direction and the predicted one become, the better performance RNN will achieve, which ignores the preferences beyond the label. In addition, there is no work on generating recommendations with *user preference direction* and providing explanations in serendipity recommendation. In short, there are three open challenges on directional and explainable serendipity recommendation: (1) *what* are the users' preference directions; (2) *how* to generate serendipitous recommendations with *user preference direction*; (3) explaining *why* the items are recommended.

Our Motivations. To address the above challenges, we have three motivations. (1) *Set users' preference directions.* We manage to infer users' various long-term preferences and short-term demands as the basis of *user preference direction* in an explainable way. (2) *Generate serendipitous recommendations.* We try to combine user-item relevance with *user preference direction* to recommend items. (3)*Provide explanations.* We strive to offer reasonable explanations based on users' histories.

We propose **DESR**, which is composed of four parts: inferring long-term preferences, capturing short-term demands, generating recommendations, and providing explanations. The key contributions of this work are as follows:

(1) *Preference extraction with GMM and demand capture with the capsule network.* We adopt the Gaussian Mixture Model and the capsule network to capture users' long-term preferences and short-term demands, respectively. (Sections 3.1 and 3.2)

(2) *Serendipity recommendation with serendipity vector.* We integrate item-user distances and users' preferences as the proposed *serendipity vector* and utilize it to recommend items, strengthening the role of *user preference direction.* (Sections 2.1 and 3.3)

(3) *Back-routing for explanations.* We make the first attempt to provide explanations on serendipitous recommendations, increasing users' trust and acceptance. (Section 3.4)

(4) *Proposal for novel fine-grained metrics.* We define some novel metrics based on the embedding representations, so as to distinguish between paradoxically equal recommendations. Especially *AD*, a metric to summarize accuracy and difference, plays a vital role in directly measuring the serendipity. (Section 4.1)

(5) *Improvements on overall performance.* We conduct comprehensive experiments on real-world datasets: MovieLens-1m[2] [12] and Amazon-Kindle-Store[3] [13, 27]. Compared with existing methods, **DESR** achieves better performance on serendipity, the balance between accuracy and difference. E.g., it improves *AD* by 29.3% on MovieLens-1m and 31.53% on Amazon-Kindle-Store, compared with **HAES**, a recent serendipity-oriented method. (Section 4.3)

2 PROBLEM STATEMENT AND MODEL OVERVIEW

In this section, we define the key concepts and the problem we solve. Then, we provide a brief overview of our solution. Notations are described in Table 1, where we name a preference representation as *preCap* referring to the definition of capsule[4] in [15]. It is the same for *reCap* and *expCap*.

2.1 The Key Concepts

The formal definitions of *user preference direction* and *serendipity vector* are presented as follows.

Definition 1: User Preference Direction. Given an embedding space, we set the preference direction as from a short-term demand (denoted as *deCap*) to a long-term preference (denoted as *preCap*). We define that the items on any preference direction of u_{tar} meet *user preference direction*, whose formal definition is given by the proposed *UPD* function. An item, t_i, meets *User Preference*

[2]https://grouplens.org/datasets/movielens/1m/
[3]http://jmcauley.ucsd.edu/data/amazon/
[4]A capsule is a group of neurons whose outputs represent different properties of the same entity.

Table 1: Notations

Symbol	Description
U	users $\{u_0, u_1, \dots, u_n\}$, where u_{tar} is the target user
T	items $\{t_0, t_1, \dots, t_m\}$
M	a rating matrix with timestamps
$itemCap$	a capsule representing the input item, $t_i \in T$
$preCap$	a capsule representing the long-term preference
$deCap$	a capsule representing the short-term demand
$reCap$	a capsule representing the recommended item
$expCap$	a capsule used for the explanation

Direction when $UPD(t_i) = 1$,

$$UPD(t_i) = \begin{cases} 1 & \frac{\overrightarrow{t_i - deCap}}{\|\overrightarrow{t_i - deCap}\|} = \frac{\overrightarrow{preCap - deCap}}{\|\overrightarrow{preCap - deCap}\|} \\ 0 & otherwise \end{cases}. \quad (1)$$

Take Fig. 1(a) as an example, where t_0 meets *user preference direction*. If the optimal candidates (e.g., t_0, t_4, t_6) are not enough to generate recommendations, we pose that the items close to u_{tar}'s preference directions (e.g., t_3) also satisfy *user preference direction*.

Definition 2: Serendipity Vector. The *serendipity vector* draws optimal points for serendipitous recommendations. Its direction coincides with the corresponding *user preference direction*, and its magnitude denotes the range for serendipity recommendations.

2.2 Problem Definition

Given a user set, an item set, and a rating set, the purpose is to recommend serendipitous items that meet *user preference direction* and offer a reasonable explanation for each item recommended.

Input. The input is an initial user-item record set, I (U, T, M), and a target user, u_{tar}, where U is a user set, T is an item set denoted by $itemCaps$, and M is a rating matrix with timestamps.

Output. The goal is to generate a list of potential items, $T_{tar} \subset T$, each one of which is represented as a *reCap*, and provide explanations for u_{tar}, so as to enhance recommendation serendipity (a balance between difference and accuracy) and explainability.

Objective.

$$Maximize \quad serendipity(T_{tar}, u_{tar}), \quad (2)$$

where $\forall reCap \in T_{tar}$ meets *user preference direction*.

2.3 Model Overview

We propose a directional and explainable serendipity recommendation approach named **DESR**, whose framework is illustrated in Fig. 2. (1) *Long-term Preferences Extraction.* The component tries to infers users' long-term preferences as *preCaps*, with an unsupervised method based on GMM. (2) *Short-term Demands Capture.* The capsule network is exploited to capture users' short-term demands, *deCaps*, in consideration of its success on representation and explainability. (3) *Recommendations Generation.* Based on *preCaps* and *deCaps*, we calculate the *serendipity vector* and generate recommendations (denoted as *reCaps*). (4) *Back-routing for Explanations.* A back-routing strategy from *reCaps* to *itemCaps* is employed to provide explanations for u_{tar}.

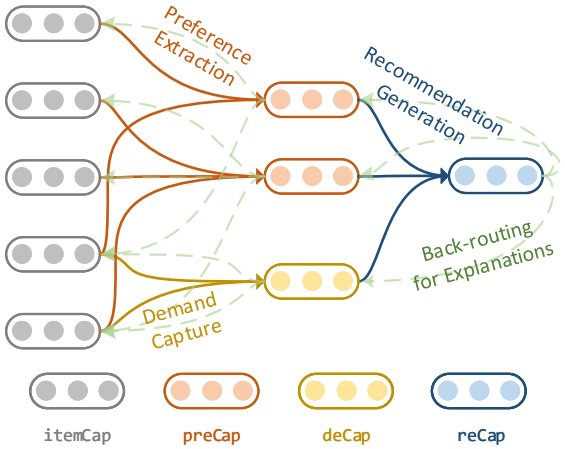


Figure 2: The framework of DESR.

3 DESR: THE DETAILS

In this section, we introduce the details of each component in **DESR**: long-term preferences extraction, short-term demands capture, recommendation generation, and back-routing for explanations.

3.1 Long-term Preferences Extraction

We give a brief introduction of the Gaussian mixture model and exploit it to develop a comprehensive representation of users' long-term preferences based on all their historical items.

3.1.1 Gaussian Mixture Model. GMM (Gaussian mixture model) [33] is extensively used in clustering analysis, which could be viewed as a linear combination of multiple Gaussian distributions. We define GMM, $P(y|\theta)$, that we employ in this paper, as follows,

$$P(y|\theta) = \sum_{k=1}^{K} \alpha_k \phi(y|\theta_k), \quad (3)$$

$$\phi(y|\theta_k) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp(-\frac{(y-\mu_k)^2}{2\sigma_k^2}), \quad (4)$$

where α_k is a non-negative mixture weight, $\sum_{k=1}^{K} \alpha_k = 1$, which represents the significance of $\phi(y|\theta_k)$, and $\phi(y|\theta_k)$ is the k-th Gaussian density function. $\theta_k = (\mu_k, \sigma_k^2)$ is the parameter of $\phi(y|\theta_k)$, which respectively denotes its mean and square. In the clustering analysis with GMM, μ_k denotes the k-th clustering center of samples like y, and σ_k^2 represents the corresponding clustering level.

The process that a large amount of discrete points converge into K clusters with GMM, is similar to the process that lots of items rated by u_{tar} are abstracted into multiple different preferences.

3.1.2 Preference Extraction. Users' preferences are usually predicted by RNN as a single vector, which couldn't reflect the various aspects of the long-term preferences. Motivated by the similarity between the two processes (i.e., clustering and preference extraction) and the success of GMM on clustering, we employ it to comprehensively represent users' long-term preferences.

First, we represent items as vectors and group them by users. Then, for each user, we adopt the grouped items as the input of GMM and utilize the EM algorithm [34] to determine the parameters

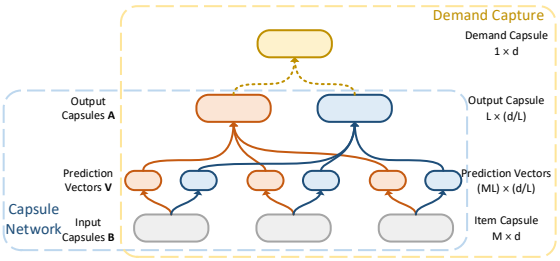


Figure 3: The process of the capsule network.

in Eq. 3. Finally, similar items are gathered in a cluster, represented by a Gaussian function (see Eq. 4), where μ_k denotes the k-th long-term preference of the corresponding user.

3.2 Short-term Demands Capture

We briefly review the capsule network and describe the process of capturing short-term demands with recent related items, which is also an indispensable task in the directional recommendation.

3.2.1 The Capsule Network. A capsule is a group of neurons whose outputs represent various properties of the particular entity [14, 35], alleviating the limitation on the representation of CNN and RNN. As is shown in Fig. 3, the capsules at the higher level, A, are the abstraction of capsules at the lower level, B. In the abstraction process, an iterative routing-by-agreement algorithm [35] is used: a capsule, $b_i \in B$, prefers to send its output to $a_j \in A$ when the "prediction vector" of b_i is more similar to a_j. With the processing, similar lower-level capsules tend to gather as a higher-level capsule, which could have an overall representation of lower-level capsules.

The part-to-whole process is similar to the process of item-to-demand in the recommender systems. It inspires us to adopt the capsule network to capture users' short-term demands.

3.2.2 Demand Capture. Fig. 3 displays the process of capturing short-term demands, where M items (denoted as *itemCaps*) are employed as the input capsules to predict a demand capsule, *deCap*. The transformation from *itemCaps* to output capsules is completed with the routing-by-agreement scheme [35]. Different from the original capsule network [35], there are L output capsules, each one of which is a $\frac{d}{L}$-dimension vector representing a distinct characteristic. We concentrate all output capsules as a predicted *deCap*.

In training of the capsule network, there are two major tasks, data pre-process and loss function definition. (1) *Data Pre-process.* We group items by users and sort all items within groups in chronological order to generate the input. For each sorted item list, M items (denoted as *itemCaps*) are selected as the input, and the next one is set as the label. (2) *Loss Function.* Our goal is to make the predicted demand, $\hat{deCap}_i$, and the labeled demand, $deCap_i$, as similar as possible. Hence, MSE (Mean Squared Error) is adopted as the loss function:

$$loss = \frac{1}{n}\sum_{i=1}^{n}(deCap_i - \hat{deCap}_i)^2, \tag{5}$$

where n is the number of training samples.

3.3 Recommendation Generation

Based on long-term preferences and short-term demands, we determine the *serendipity vector* and generate recommendations.

3.3.1 Serendipity Vector Calculation. The *serendipity vector* plays a guiding role in **DESR**, determining the location for serendipitous recommendations in the embedding space. Its calculation is divided into two steps: determining the magnitude and setting the direction.

Determining the magnitude. Users usually know items near the *deCap* (a capsule to denote short-term demand) well and have no interest in items far away from it. Between the two, there would be an area containing serendipitous items (i.e., *reCaps*), which are not apparently familiar for users but meet their short-term demands. It is crucial to find a suitable distance between the *deCap* and *reCap* (see Fig. 4) – the magnitude of *serendipity vector*.

Suppose (1) users with multiple preferences tend to accept more different items and (2) they are more familiar with items on the preference direction which involves more historical items. Then, the magnitude of the *serendipity vector* would be related to the scope of the long-term preferences, $S(u_{tar})$, and the familiarity for *preCap* (a capsule to represent a long-term preference), $F(preCap)$. We quantify $S(u_{tar})$ and $F(preCap)$ as follows,

$$S(u_{tar}) = f_S(K) \tag{6}$$

$$F(preCap) = f_F(|\{t_1, t_2, ..., t_p\}|), \tag{7}$$

where u_{tar} has K long-term preferences and $\{t_1, t_2, ..., t_p\}$ is an item cluster corresponding to *preCap*. f_S and f_F are to normalize K and $|\{t_1, t_2, ..., t_p\}|$ to a suitable range, $(0, lim_S)$ and $(0, lim_F)$ respectively. Based on the quantification of $S(u_{tar})$ and $F(preCap)$, we estimate the magnitude of *serendipity vector*, as follows,

$$\|\overrightarrow{serendipity\ vector}\| = m_{base}(1 + S(u_{tar}))(1 + F(preCap)), \tag{8}$$

where m_{base} denotes the minimum distance to find serendipitous items for all users, which would float on different datasets.

Setting the direction. Besides determining a suitable magnitude, a proper direction of *serendipity vector* would also contribute to serendipitous recommendation by cutting numerous unrelated items down. Users usually prefer items with particular characteristics, as observed in Fig. 1(c), which illustrates that the extension from *deCap* should favor to users' preferences. It motivates us to set the direction of *serendipity vector* as *user preference direction* – from the short-term demand to long-term preference, as follows,

$$\frac{\overrightarrow{serendipity\ vector}}{\|\overrightarrow{serendipity\ vector}\|} = \frac{\overrightarrow{preCap - deCap}}{\|\overrightarrow{preCap - deCap}\|}. \tag{9}$$

3.3.2 Recommendation Generation. *Serendipity vectors* determine the location to generate serendipitous items, and then we would allocate recommendations to each of them. The sampling ratio on the i-th *serendipity vector*, s_i, is calculated as follows,

$$s_i = \frac{T_i}{\sum_{i=1}^{K} T_i}, \tag{10}$$

where T_i is the number of items related to the corresponding preference, and there are K preferences of u_{tar}. For each user, we would recommend L_R items, of which there are $s_i * L_R$ items generated based on i-th *serendipity vector*.

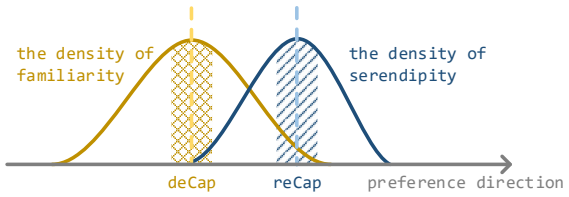


Figure 4: The intuition on the magnitude of *Serendipity Vector*.

Table 2: Statistics of datasets.

Item	Statistic	
	MovieLens-1m	Amazon-Kindle-Store
# users	6040	3061
# items	3260	6073
# ratings	998539	132594
density [5]	5.07%	0.71%

3.4 Back-routing for Explanations

To increase users' acceptance, a back-routing scheme, from *reCap*s to *itemCap*s, is exploited to provide explanations in this subsection.

For each item recommended, $reCap_{tar}$, we would select a *preCap* or *deCap* related to u_{tar}, which is the most similar to $reCap_{tar}$, as the basic for explanations, *expCap*,

$$expCap = Minimize\ dis(cap, reCap_{tar}), \tag{11}$$

where $cap \in \{deCap,\ preCap_1,\ preCap_2,\ \dots,preCap_K\}$. A list of items related to *expCap*, $\{t_1,\ t_2,\ \dots,\ t_E\}$, are selected to generate explanations. There are two types of explanations: (1) $explanation_1$: "The item is similar to the items, $\{t_1,\ t_2,\ \dots,\ t_E\}$, which you watched for a long time." (2) $explanation_2$: "The item is similar to the items, $\{t_1,\ t_2,\ \dots,\ t_E\}$, which you recently watched.". We select explanations based on the type *expCap*, as follows,

$$explanation = \begin{cases} explanation_1 & \text{expCap}\in\text{preCaps} \\ explanation_2 & \text{expCap}\in\text{deCaps} \end{cases}. \tag{12}$$

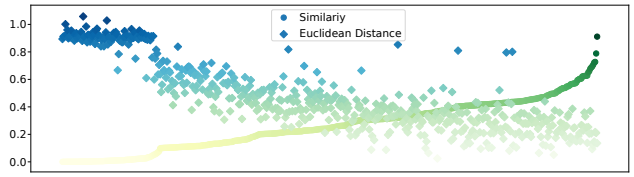
4 EXPERIMENTS AND RESULTS

We conduct comprehensive experiments on real-world datasets to comparatively evaluate and demonstrate the effectiveness of the proposed method. The experiment section is organized as follows. We (1) introduce basic experiment settings, including datasets, baselines, and metrics; (2) verify the effects of parameters in the proposed metrics and model; (3) compare our method with benchmarks to illustrate its improvements; (4) verify the effectiveness of components; and (5) show a case study for preference extraction and explanation generation.

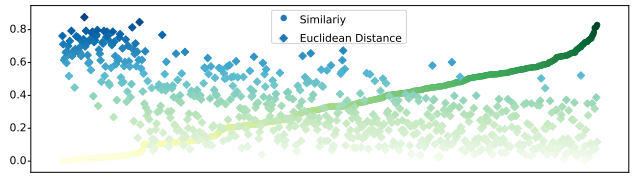
4.1 Experiment Settings

We develop the experiments on real-world datasets — MovieLens-1m (ml-1m) [12] and Amazon-Kindle-Store (Kindle) [13, 27] — where we take the top 80% ratings as the training set and the rest as the testing set according to the timestamps of ratings. The statistics of datasets are shown in Table 2. We introduce data preprocessing, baselines, and evaluation metrics.

[5] the density of the rating matrix, $density = \frac{|S|}{|U|*|T|}$



(a) ml-1m



(b) Kindle

Figure 5: Visualization of item similarity and node distances in the embedding space, where a circle represents a item-item similarity factor, and a square denotes a node-node Euclidean distance in the embedding space.

4.1.1 Data Preprocessing. In the experiments, we exploit proNE [41] to represent items as embedding nodes, which is expected to embed similar items closely in the embedding space. Its performance is shown in Fig. 5, from which we find that there is usually a smaller distance between similar items.

4.1.2 Baselines. We select a widely used serendipity-oriented method, **KFN** [36], and a recent approach named **HAES** [24] as baselines. In addition, considering **DESR** tries to make a trade-off between difference and accuracy, a random method, **RAND**, and an accuracy-based approach, $\mathbf{ACC}_{LSTM}$, are also adopted as benchmarks.

RAND. The random-based method randomly generates recommendations from items which u_{tar} hasn't rated in the history.

$\mathbf{ACC}_{LSTM}$**.** The accuracy-based recommendation is one of the most widely used personalized recommendation methods. It recommends movies consistent with users' past behaviors. LSTM (Long Short-Term Memory) is used as the basis of $\mathbf{ACC}_{LSTM}$ in the paper.

KFN. The intention of **KFN** is similar to that of KNN, which is to offer recommendations based on neighbors' behaviors. **KFN** recommends items that dissimilar users dislike to improve serendipity.

HAES. HAES adaptively adjusts the level of relevance between recommendations and u_{tar} with an elastic strategy, to reduce unrelated recommendations and improve recommendation serendipity.

4.1.3 Metrics. Recent years have seen significant development in the embedding techniques and rapid growth in the number of items, which calls for some fine-grained metrics to more accurately evaluate the recommendation performance.

For instance, if t_i and t_j are recommended but neither of them have rated by u_{tar}, they will be taken as equally negative results. However, in the embedding space, if there is a smaller distance between t_i and u_{tar}, it will indicate that u_{tar} is more familiar with t_i. Hence, we propose some fine-grained metrics based on the node-node distances in the embedding space, to distinguish some paradoxically equal results (e.g., t_i, t_j). First, we measure the

Table 3: Metrics

Metric	Abbreviation	Description
accuracy (Eq. 16)	-	a metric to measure the recommendation accuracy
accuracy_preference (Eq. 17)	*acc_pre*	a metric to measure the similarity of *reCap*s and *preCap*s
accuracy_demand (Eq. 18)	*acc_de*	a metric to measure the similarity of *reCap*s and *deCap*s
accuracy_all (Eq. 19)	*acc*	a metric to summarize *accuracy*, *acc_pre* and *acc_de*
diversity (Eq. 20)	*div*	a metric to measure the diversity of recommendations
difference (Eq. 21)	-	a metric to measure the difference between recommendations and u_{tar}'s history
difference_all (Eq. 22)	*dif*	a metric to summarize *diversity* and *difference*
AD (Eq. 23)	-	a metric to comprehensively measure *acc* and *dif*

distance between $node_i$ and $node_j$, as follows,

$$dis_{ori}(node_i, node_j) = f_{dis}(emb_i, emb_j), \tag{13}$$

$$dis(node_i, node_j) = \frac{dis_{ori}(node_i, node_j)}{dis_{max}}. \tag{14}$$

emb_i is the embedding vector of $node_i$, dis_{ori} is the original distance in the embedding space between $node_i$ and $node_j$ calculated by f_{dis} (e.g, euclidean distance, cosine distance, etc.). Eq. 14 is used to normalize dis_{ori}. If the euclidean distance function is used as f_{dis}, f_{dis} will be positively correlated to MSE, mean square error.

$$f_{dis}(emb_i, emb_j) = \sqrt{d * MSE(emb_i, emb_j)}, \tag{15}$$

where d is the dimension of embedding vectors, and the euclidean distance function is employed as f_{dis} in the paper. We propose some specific metrics (shown in Table 3) based on the *dis* function.

Accuracy. We define *accuracy* to access the recommendation performance based on the recommended results and labeled items.

$$\begin{aligned} accuracy &= 1 - \frac{1}{L_R}\sum_{i=1}^{L_R} dis(t_i, t_{label}) \\ &= 1 - \frac{1}{L_R}\sum_{i=1}^{L_R} min(\{dis(t_i, t_1), dis(t_i, t_2) \\ &\quad \ldots dis(t_i, t_{L_{label}})\}), \end{aligned} \tag{16}$$

where L_R items are recommended for u_{tar}, and there are L_{label} labeled items of u_{tar} in the testing set. For each recommended item, t_i, the distance between it and the labeled item, $dis(t_i, t_{label})$, is the minimum value of all distances between t_i and all labeled items.

Acc_pre. *Accuracy_preference* (*acc_pre*) is defined to quantify the similarity between recommendations and u_{tar}'s long-term preferences, *preCap*s. Given a recommended item, t_i, where $t_i \in \{t_1, t_2, ..., t_{L_R}\}$, we define *acc_pre* of recommendations, as follows,

$$\begin{aligned} acc\_pre &= 1 - \frac{1}{L_R}\sum_{i=1}^{L_R} dis(t_i, preCap_{tar}) \\ &= 1 - \frac{1}{L_R}\sum_{i=1}^{L_R} min(\{dis(t_i, preCap_1), dis(t_i, preCap_2) \\ &\quad \ldots dis(t_i, preCap_K)\}). \end{aligned} \tag{17}$$

Acc_de. *Accuracy_demand* (*acc_de*) is to quantify the similarity between recommendations and u_{tar}' short-term demand, $deCap_{tar}$,

$$acc\_de = 1 - \frac{1}{L_R}\sum_{i=1}^{L_R} dis(t_i, deCap_{tar}). \tag{18}$$

Accuracy_all. *Accuracy_all* (*acc*) is defined to summarize *accuracy*, *acc_pre* and *acc_de* as a whole, where η and θ represent the weights of *acc_pre* and *acc_de*, respectively.

$$acc = \frac{1}{1+\eta+\theta}(accuracy + \eta acc\_pre + \theta acc\_de). \tag{19}$$

Diversity. We calculate the diversity of recommendations, *diversity* (*div*), based on the distance between them in the embedding space,

$$div = \frac{1}{L_R * L_R}\sum_{i=1}^{L_R}\sum_{j=1}^{L_R} dis(t_i, t_j). \tag{20}$$

Difference. *Difference* denotes the difference between recommendations (R) and users' histories (H), which is defined as follows,

$$difference = \frac{1}{L_R * L_H}\sum_{i=1}^{L_R}\sum_{j=1}^{L_H} dis(t_i, t_j), \tag{21}$$

where there are L_H items rated by u_{tar} in the history.

Difference_all. *difference_all* (*dif*) is to sum *div* and *difference* up, where λ and γ denote the weights of *div* and *difference*,

$$dif = \frac{1}{\lambda+\gamma}(\lambda div + \gamma difference). \tag{22}$$

AD. Considering the objectives of serendipity recommendation - both accuracy and difference - and the collision between *acc* and *dif*, a metric that could comprehensively measure them is required. The intuition of *AD* is similar to that of $F-score$, balancing *precision* and *recall*, which inspires us to define a metric, *AD*, as follows,

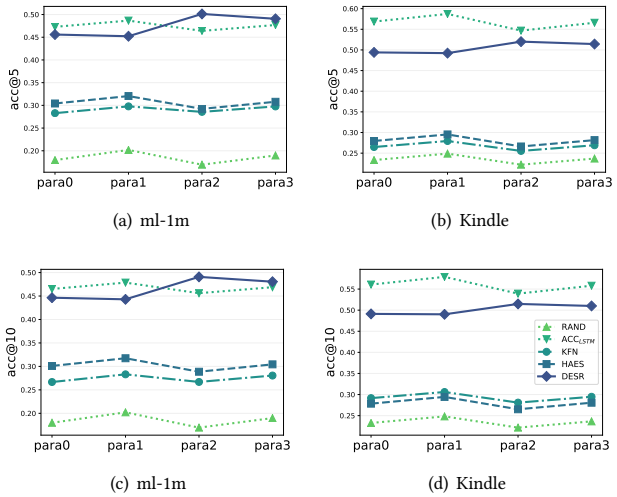
$$AD = \frac{acc * dif}{acc + dif}. \tag{23}$$

4.2 Effects of Parameters

To enhance the flexibility of the proposed metrics and model, some parameters are established. In this section, we vary these parameters to investigate their effects on the performance.

4.2.1 Varying the parameters in metrics. There are lots of parameters (i.e., η, θ, λ, γ) in our definitions for *acc* and *dif*, which make a direct impact on the access of recommendation performance.

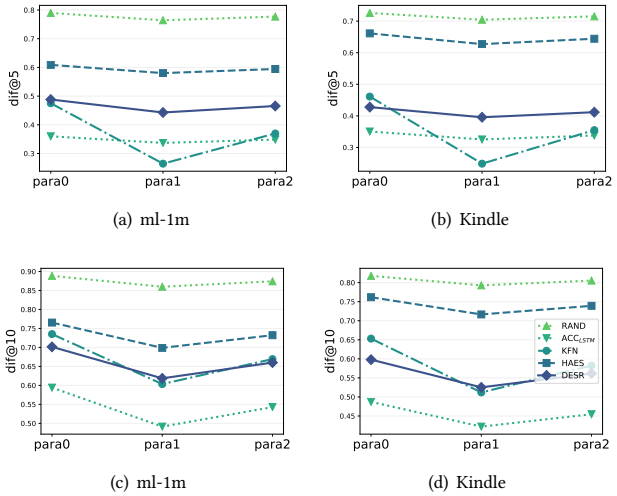
To check the effects of different parameters on *acc*, we successively set (η, θ) as (0.5, 0.5), (0.5, 1), (1, 0.5), and (1, 1), to assign



(a) ml-1m (b) Kindle

(c) ml-1m (d) Kindle

Figure 6: Effects of parameters on acc. (para0: $\eta = 0.5, \theta = 0.5$; para1: $\eta = 0.5, \theta = 1$; para2: $\eta = 1, \theta = 0.5$; para3: $\eta = 1, \theta = 1$.)



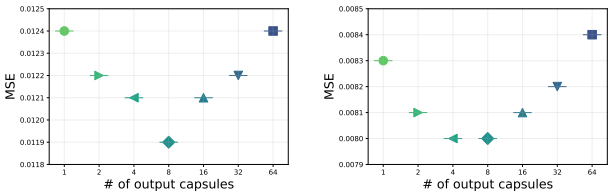
(a) ml-1m (b) Kindle

(c) ml-1m (d) Kindle

Figure 7: Effects of parameters on dif. (para0: $\lambda = 0.5, \gamma = 1$; para1: $\lambda = 1, \gamma = 0.5$; para2: $\lambda = 1, \gamma = 1$.)

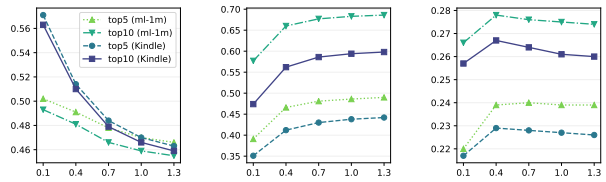
different weights to *acc_pre* and *acc_de*. We find that varying parameters exerts distinctly different impacts on the performance for most approaches, especially $\mathbf{ACC}_{LSTM}$ and **DESR**. For example, given the top 5 recommendations on MovieLens-1m (see Fig. 6(a)), **DESR** with a bigger weight on *acc_pre* improves *acc* by 5.69%, while it decrease *acc* by 10.49% when a bigger weight is assigned to *acc_de*, compared with $\mathbf{ACC}_{LSTM}$. A bigger weight of *acc_pre* (i.e., $(\eta, \theta) = (1, 0.5)$) makes **DESR** outperform $\mathbf{ACC}_{LSTM}$ on *acc*. That is because that our method is even more in line with users' long-term preferences, with the help of GMM (see Table 4). As a comparison, $\mathbf{ACC}_{LSTM}$ (based on LSTM) is better at capturing users' current preferences instead of comprehensive preferences, which is related to the forget gate of LSTM.

To verify the effects of parameters on dif, we successively set (λ, γ), the weight of $(div, difference)$, as (0.5, 1), (1, 0.5), and (1, 1).



(a) ml-1m (b) Kindle

Figure 8: Effects of the number of output capsules.



(a) acc (b) dif (c) AD

Figure 9: Effects of m_{base}

An interesting finding is gained: serendipity-oriented methods (i.e., **KFN**, **HAES**, **DESR**) perform better on dif when a bigger weight is assigned to $difference$, since they generate recommendations which have a bigger difference with users' histories. For example, $difference$ is 38.71% bigger than div when the top 5 items are recommended on MovieLens-1m. In addition, we also find that overall, our method performs poorer than **HAES** on dif, which demonstrates that *user preference direction* brings a bigger possibility of reducing unrelated items.

Considering the equal need for *acc_pre* and *acc_de* on *acc*, div and $difference$ on dif, we set each parameters (i.e., η, θ, λ, γ) as 1.

4.2.2 Varying the parameters in DESR. Some parameters are set in our method, L and m_{base}, where L denotes the number of output capsules in Section 3.2) and m_{base} (see Eq. 8) determines the minimum magnitude of the *serendipity vector*. In this paper, we set lim_S=0.1 and lim_F=0.2 (see Eqs. 6 and 7).

To determine the suitable number of output capsules, we set it as 1, 2, 4, 8, 16, 32, 64 respectively, whose comparison on MovieLens-1m is shown in Fig. 8(a) and that on Amazon-Kindle-Store is shown in Fig. 8(b). We find that it is hardly possible to capture accurate short-term demands when L is too big or small, e.g., the capsule network has the worst performance when the maximum (i.e., 64) or the minimum (i.e., 1) is assigned to the number of output capsules (see Fig. 8). This is related to the function of output capsules, abstracting the features of $itemCaps$ for $decap$, where L determines the number of features. Setting L as 64 means determining a feature on each dimension, while assigning 1 to L indicates that only one feature is abstracted. According to the results, we find 8 as the optimal value for L and we set L as 8 in the following experiments.

In addition, to improve the performance of **DESR**, it's necessary to find a suitable value as m_{base}. Since the biggest node-node distance in our embedding space is about 1.5, we vary the value of m_{base} from 0.1 to 1.3 with an interval of 0.3 to check its effects on recommendation. The results are shown in Fig. 9. We find that by increasing of m_{base}, the performance on *acc* becomes poorer and

Table 4: Comparison on original metrics. (ml-1m)

method	accuracy@5	accuracy@10	acc_pre@5	acc_pre@10	acc_de@5	acc_de@10	div@5	div@10	difference@5	difference@10
RAND	0.15	0.15	0.128	0.129	0.291	0.291	**0.739**	**0.831**	**0.819**	**0.918**
ACC$_{LSTM}$	**0.461**	**0.454**	0.428	0.42	**0.541**	**0.533**	0.314	0.389	0.383	0.697
KFN	0.238	0.225	0.297	0.268	0.358	0.349	0.054	0.472	0.686	0.867
HAES	0.293	0.291	0.245	0.239	0.386	0.383	0.551	0.632	0.638	0.832
DESR	0.352	0.344	**0.683**	**0.668**	0.437	0.429	0.398	0.536	0.533	0.784

Table 5: Comparison on original metrics. (Kindle)

method	accuracy@5	accuracy@10	acc_pre@5	acc_pre@10	acc_de@5	acc_de@10	div@5	div@10	difference@5	difference@10
RAND	0.222	0.221	0.177	0.177	0.311	0.31	**0.683**	**0.768**	**0.748**	**0.843**
ACC$_{LSTM}$	**0.577**	**0.569**	0.459	0.453	**0.661**	**0.652**	0.3	0.357	0.375	0.552
KFN	0.252	0.282	0.218	0.238	0.338	0.364	0.036	0.371	0.674	0.794
HAES	0.273	0.272	0.213	0.212	0.358	0.358	0.593	0.671	0.695	0.807
DESR	0.433	0.435	**0.624**	**0.609**	0.485	0.486	0.363	0.452	0.46	0.671

Table 6: Comparison on overall recommendation (ml-1m).

method	acc@5	acc@10	dif@5	dif@10	AD@5	AD@10
RAND	0.19	0.19	**0.777**	**0.874**	0.152	0.156
ACC$_{LSTM}$	0.477	0.469	0.348	0.543	0.201	0.252
KFN	0.298	0.28	0.37	0.669	0.165	0.198
HAES	0.308	0.304	0.595	0.732	0.203	0.215
DESR	**0.491**	**0.48**	0.466	0.66	**0.239**	**0.278**

Table 7: Comparison on overall recommendation (Kindle).

method	acc@5	acc@10	dif@5	dif@10	AD@5	AD@10
RAND	0.237	0.236	**0.715**	**0.805**	0.178	0.183
ACC$_{LSTM}$	**0.566**	**0.558**	0.338	0.455	0.212	0.25
KFN	0.269	0.295	0.355	0.582	0.153	0.196
HAES	0.282	0.281	0.644	0.739	0.196	0.203
DESR	0.514	0.51	0.412	0.562	**0.229**	**0.267**

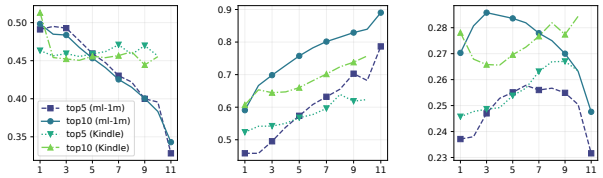
the performance on dif becomes better. The reason for this is that the larger m_{base} grows, the more unfamiliar the target user and recommendations become. Another finding is that between the two extremes - the most accurate recommendations and the most different ones - there is balance point representing the serendipitous ones. When $m_{base} = 0.4$, our approach reaches a balance point and we set m_{base} as 0.4 in the following experiments.

4.3 Overall Comparison

We compare **DESR** with baselines to verify its effectiveness on recommendation serendipity and show its promotion for diversity, based on the top 5 and 10 recommendations, respectively. The comparison on original metrics is illustrated in Tables 4 and 5.

4.3.1 Comparison on Serendipity. We compare **DESR** with baselines on *acc*, dif, and *AD*. The results on MovieLens-1m (ml-1m) and Amazon-Kindle-Store (Kindle) are shown in Tables 6 and 7.

Compared with the existing best serendipity-oriented method, **HAES**, (1) for top 5 recommendations, **DESR** improves *AD* by 17.73% on MovieLens-1m and 16.84% on Amazon-Kindle-Store; (2) for top 10 recommendations, **DESR** improves *AD* by 29.30% on MovieLens-1m and by 31.53% on Amazon-Kindle-Store. The enhancements demonstrate that our method achieves the best performance on both datasets. The reason is that **DESR** increases dif at a lower cost of *acc*, where it improves dif by 21.89% at the cost of decreasing *acc* by 10.12% on Amazon-Kindle-Store, and increases both *acc* and dif on MovieLens-1m(see Tables 6 and 7), compared with **ACC**$_{LSTM}$. It is closely associated with *user preference direction*. In addition, we find that our approach performs better on MovieLens-1m, which is consistent with the performance of embedding (see Fig. 5). The possible reason is that the density of rating matrix on MovieLens-1m, 5.07%, is even bigger than that on Amazon-Kindle-Store, 0.71%, as is shown in Table 2. The intensive interactions among users are beneficial to developing their relationship.



(a) acc (b) dif (c) AD

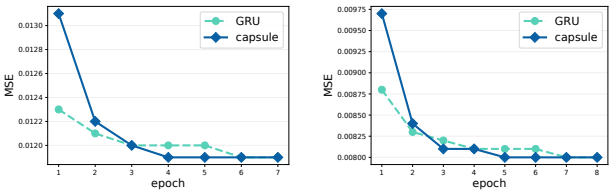
Figure 10: Comparison on different user group.

4.3.2 Comparison on Diversity. Another finding is that the more accurate on the fine-grained preferences recommendations are, the less diverse they are. However, our approach maximizes diversity under the premise of ensuring recommendation accuracy, which is illustrated in the comparison between **ACC**$_{LSTM}$ and **DESR**. It increases div by 37.79% on MovieLens-1m (see Table 4), 26.61% on Amazon-Kindle-Store (see Table 5) when the top 10 items are recommended. It indicates that *user preference direction* gives our approach a tremendous impetus to enhance recommendation diversity while preserving accuracy.

4.4 Verification on Components

After demonstrating the enhancements on overall performance, we verify the effectiveness of components, preference extraction and demand capture.

4.4.1 Verification on Preference Extraction. We illustrate the improvements achieved by preference extraction component and investigate how the number of preferences affect the performance.



(a) ml-1m (b) Kindle

Figure 11: Comparison between the capsule network and GRU on capturing users' short-term demands.

The comparison on *acc_pre* (see Tables 4, 5) **displays our improvements on preference accuracy**, where **DESR** increases *acc_pre* by at least 33.33% even when compared to the accuracy-oriented method ($\mathbf{ACC}_{LSTM}$). It illustrates that our approach could cater to users' long-term fine-grained preferences with the help of preference extraction. Another valuable finding is that our method performs even better than other serendipity-oriented methods, for example, compared with **HAES**, it improves *accuracy* by 18.21% on MovieLens-1m and by 59.93% on Amazon-Kindle-Store when the top 10 items are recommended. It demonstrates that a comprehensive preference representation could promote the recommendation system to gain a better understanding of users' behavioral patterns, which is also raised in [37].

There is a big difference between the numbers of preferences on real-world datasets, for instance, the maximum is 11 on MovieLens-1m and 10 on Amazon-Kindle-Store and the minimum is 1 on both datasets. We investigate the performance difference among various user groups with different numbers of preferences. Based on the observations in Fig. 10, we find that **DESR** tends to recommend different items to the users with more preferences, and vice versa. Another finding is that the best overall performance (*AD*) is achieved with the users whose preference number is in the middle (e.g., 6 on MovieLens-1m and 8 on Amazon-Kindle-Store when top 10 items are recommended). It is possibly related to the fact that: (1) for users with fewer preferences, there are too few historical behaviors to model users' preferences; (2) for users with many preferences, the behavioral patterns are too difficult to capture.

4.4.2 Verification on Demand Capture. We compare the capsule network with GRU (Gated Recurrent Unit, a form of RNN) which is one of the best techniques for processing sequence data, to demonstrate its superiority on demand capture.

We provide the same input for the capsule network and GRU and set a unified dimension for the output. The dimension of input is 20×64, where 20 items (each one is represented as a 64-dimension vector) are employed to predict the next item as a *deCap*. The output is a 64-dimension vector. MSE (mean square error, see Eq. 5) is adopted as the metric.

Fig. 11 depicts the comparison between the capsule network and GRU, where they reach the same optimal MSE, 0.0119 on MovieLens-1m and 0.0080 on Amazon-Kindle-Store. On MovieLens-1m (see Fig. 11(a)), the capsule network reaches the best MSE at the fourth epoch and GRU at the sixth epoch; on Amazon-Kindle-Store (see Fig. 11(b)), the capsule network reaches the best MSE at the fifth epoch and GRU at the seventh epoch. We conclude that (1) the capsule network and GRU could arrive at the same best performance on demand capture but (2) the former has a faster convergence. In addition, the capsule network outperforms GRU on explainability.

Hence, the capsule network is a better choice to capture users' short-term demands, in view of its efficiency and explainability.

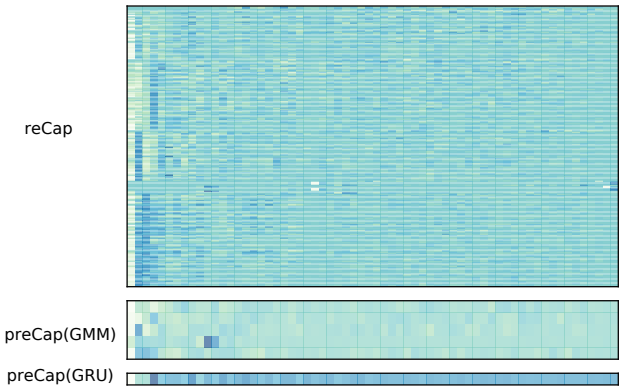


Figure 12: Comparison between GMM and GRU on capturing users' long-term preferences.

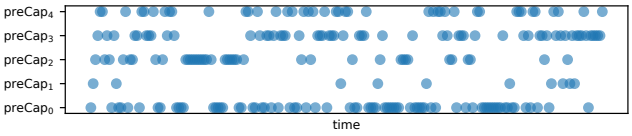


Figure 13: Distribution of item clusters by timestamps.

4.5 A Case Study

We use a case study with the same user as Fig. 1(c), a user (UserID: 4824) on MovieLens-1m, to explore the performance on preference extraction and explanation generation.

4.5.1 Case Study on Long-term Preferences. We visualize the embedding representations of items, preferences captured by GMM, and preferences predicted by GRU in Fig. 12, where for each line (an item or a preference), the color scale of each cell denotes the significance of a certain feature. The darker the cell is, the more active the corresponding item (preference) would be on the feature.

Based on the comparison between the visualizations of preferences inferred by GMM and GRU, we find that GMM could capture multiple aspects of the users' preferences, while GRU could only predict a whole preference which is similar to the labeled items (possibly similar to the items rated recently). Fig. 13 draws the item clustering distribution in chronological order, where the user's recently rated items belong to the clusters of $preCap_3$ and $preCap_4$. It encourages the preference predicted by GRU to be similar to the items in the corresponding clusters.

In short, GRU, a supervised method, optimizes parameters for a specific goal (i.e., similarity to the labeled item). However, GMM, an unsupervised model, could cluster items at first and infer preferences for multiple different individuals, strengthening the generality and integrity of users' long-term preferences.

4.5.2 Case Study on Explanations. After demonstrating the outperformance of GMM on inferring users' long-term preferences, we explore the performance on explanation generation.

Given five long-term preferences (i.e., $preCap_0$, ..., $preCap_4$), a short-term demand, *deCap*, and two recommendations, $reCap_0$,

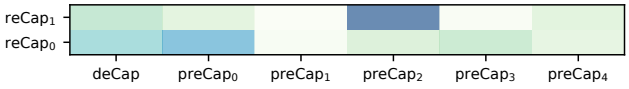


Figure 14: An illustration to show the similarity between recommendations and preferences, demands.

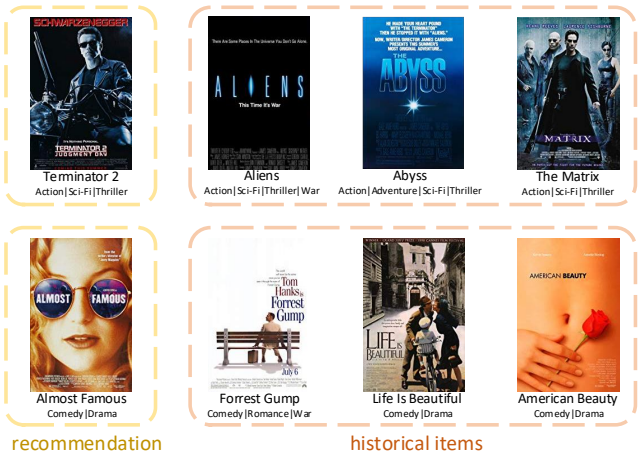


Figure 15: Selection of historical items.

$reCap_1$. Fig. 14 reveals the weights from *reCap* to *preCap* and *deCap* generated by the back-routing scheme, where the darker the cell is, the more similar *reCap* and *preCap* (*deCap*) become. We provide an explanation for each recommended item.

For $reCap_0$ (*Almost Famous*), it is more similar to $preCap_0$, where the historical items in Fig. 15 are selected in the corresponding cluster. An explanation based on $explanation_2$ (see Eq. 12) is offered, "The movie is similar to the movies, *Forrest Gump*, *Life Is Beautiful* and *American Beauty*, which you watched for a long time.". For *Terminator 2*, it is more similar to $preCap_2$, where the relevant items are also selected in the corresponding cluster and the generated explanation is similar to that of *Almost Famous*.

We find that the back-routing scheme could effectively locate historical behaviors for explanations on serendipitous items.

4.6 Summary of Experiments

In short, we have the following findings in the experiments. (1) GMM outperforms RNN on developing a comprehensive representation on long-term preferences. (2) Besides image recognition [15], capsule network also has great potential in sequence processing. (3) Keeping accuracy on users' long-term preferences would promote recommendations towards the balance between difference and accuracy, the objective of serendipity recommendation. (4) In general, **DESR** achieves a better performance on serendipity recommendation, e.g., it improves *AD* by 29.3% on MovieLens-1m and by 31.53% on Amazon-Kindle-Store, compared with **HAES**.

5 RELATED WORK

We briefly review related works on explainable recommendation and serendipity recommendation.

Explainable Recommendation. To help users accept recommendations and increase user satisfaction, some works provide explanations [1, 2, 4, 25] based on social relationships [18, 19], item features or user histories. McInerney et al. [28] propose personal explainable recommendation methods with bandits, in which they illustrate that suitable explanations (especially the ones related to users' histories) provide a significant improvement on user engagement [17]. Yu et al. [40] develop a neural attentive explainable recommender system named NAIRS, whose key component assigns attention weights to interacted items of the user. NAIRS provides explanations by displaying items with bigger attention weights.

While the above works provide reasonable explanations for accurate recommendations, to the best of our knowledge, **DESR** is the first to generate explanations in serendipity recommendation.

Serendipity Recommendation. As a solution to over-specialization, various serendipity recommendation methods are proposed to generate recommendations beyond users' bubbles. Some of them generate serendipitous recommendations through re-ranking or combining items recommended by accuracy-oriented methods [20, 22, 42]. The others try to recommend serendipitous items with novel approaches, such as a method with transfer learning [32], a curiosity-theory-based method [26], and an elasticity-driven approach [24]. Most methods focus on improving the difference between recommendations and the target user, so as to enhance recommendation serendipity, causing unrelated recommendations. To ease the limitation, some methods [24, 26] consider the users' ability to accept serendipitous recommendations. However, none of them pay attention to the guiding role of users' long-term preferences.

Our approach, **DESR**, generates serendipitous recommendations towards users' long-term preferences, which balances the accuracy and difference. The proposed user-preference-aware expansion is the first one in the serendipity recommendation, benefiting for cutting unrelated recommendations back.

6 CONCLUSIONS

In this paper, we propose a directional and explainable serendipity recommendation method **DESR**. Our main contributions are the reinforcement of *user preference direction* and explainability in serendipity recommendation. Firstly, we employ GMM to represent users' long-term preferences, in which the unsupervised method mitigates the lack of labeled data and improves the generality and integrity of the extracted preferences. Then we exploit the capsule network to capture users' short-term demands, laying a foundation for explainability. Finally, we propose a back-routing scheme to provide explanations for users, so as to increase users' trust and acceptance. Without the help of additional information (e.g., user reviews, item descriptions), the offered explanations are limited to display related items. In the future, we would like to provide more user-friendly explanations in the serendipity recommendation, to help users connect serendipitous items with their potential preferences and demands. We are also interested in applying our model to more fields, e.g., course recommendation in e-learning.

ACKNOWLEDGMENTS

This research was supported by NSFC grant 61632009, Guangdong Provincial NSF Grant 2017A030308006, Open project of Zhejiang Lab 2019KE0AB02, and in part by NSF grants CNS 1824440, CNS 1828363, CNS 1757533, CNS 1618398, CNS 1651947, and CNS 1564128.

REFERENCES

[1] Krisztian Balog, Filip Radlinski, and Shushan Arakelyan. 2019. Transparent, Scrutable and Explainable User Models for Personalized Recommendation. In *SIGIR 2019*. ACM, 265–274. https://doi.org/10.1145/3331184.3331211

[2] Jingwu Chen, Fuzhen Zhuang, Xin Hong, Xiang Ao, Xing Xie, and Qing He. 2018. Attention-driven Factor Model for Explainable Personalized Recommendation. In *SIGIR 2018*. ACM, 909–912. https://doi.org/10.1145/3209978.3210083

[3] Li Chen, Yonghua Yang, Ningxia Wang, Keping Yang, and Quan Yuan. 2019. How Serendipity Improves User Satisfaction with Recommendations? A Large-Scale User Evaluation. In *WWW 2019*. ACM, 240–250. https://doi.org/10.1145/3308558.3313469

[4] Xu Chen, Yongfeng Zhang, and Zheng Qin. 2019. Dynamic Explainable Recommendation Based on Neural Attentive Models. In *AAAI 2019*. AAAI Press, 53–60. https://aaai.org/ojs/index.php/AAAI/article/view/3768

[5] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR* abs/1412.3555 (2014). http://arxiv.org/abs/1412.3555

[6] Zeyu Cui, Zekun Li, Shu Wu, Xiaoyu Zhang, and Liang Wang. 2019. Dressing as a Whole: Outfit Compatibility Learning Based on Node-wise Graph Neural Networks. In *WWW 2019*. ACM, 307–317. https://doi.org/10.1145/3308558.3313444

[7] Yunqi Dong and Wenjun Jiang. 2019. Brand purchase prediction based on time-evolving user behaviors in e-commerce. *Concurrency and Computation: Practice and Experience* 31, 1 (2019). https://doi.org/10.1002/cpe.4882

[8] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Yihong Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation. In *WWW 2019*. ACM, 417–426. https://doi.org/10.1145/3308558.3313488

[9] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. 2010. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *RecSys 2010*. ACM, 257–260. https://doi.org/10.1145/1864708.1864761

[10] Felix A. Gers, Jürgen Schmidhuber, and Fred A. Cummins. 2000. Learning to Forget: Continual Prediction with LSTM. *Neural Computation* 12, 10 (2000), 2451–2471. https://doi.org/10.1162/089976600300015015

[11] Mihajlo Grbovic and Haibin Cheng. 2018. Real-time Personalization using Embeddings for Search Ranking at Airbnb. In *SIGKDD 2018*. ACM, 311–320. https://doi.org/10.1145/3219819.3219885

[12] F. Maxwell Harper and Joseph A. Konstan. 2016. The MovieLens Datasets: History and Context. *TiiS* 5, 4 (2016), 19:1–19:19. https://doi.org/10.1145/2827872

[13] Ruining He and Julian J. McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In *WWW 2016*. ACM, 507–517. https://doi.org/10.1145/2872427.2883037

[14] Geoffrey E. Hinton, Alex Krizhevsky, and Sida D. Wang. 2011. Transforming Auto-Encoders. In *ICANN 2011 (Lecture Notes in Computer Science)*, Vol. 6791. Springer, 44–51. https://doi.org/10.1007/978-3-642-21735-7_6

[15] Geoffrey E. Hinton, Sara Sabour, and Nicholas Frosst. 2018. Matrix capsules with EM routing. In *ICLR 2018*. https://openreview.net/forum?id=HJWLfGWRb

[16] Wenjun Jiang, Guojun Wang, Md. Zakirul Alam Bhuiyan, and Jie Wu. 2016. Understanding Graph-Based Trust Evaluation in Online Social Networks: Methodologies and Challenges. *ACM Comput. Surv.* 49, 1 (2016), 10:1–10:35. https://doi.org/10.1145/2906151

[17] Wenjun Jiang, Jie Wu, Feng Li, Guojun Wang, and Huanyang Zheng. 2016. Trust Evaluation in Online Social Networks Using Generalized Flow. *IEEE Transactions on Computers (TC)* 65(3) (2016), 952–963.

[18] Wenjun Jiang, Jie Wu, and Guojun Wang. 2015. On Selecting Recommenders for Trust Evaluation in Online Social Networks. *ACM Transactions on Internet Technology* 15, 4, Article 14 (Nov. 2015), 21 pages. https://doi.org/10.1145/2807697

[19] Wenjun Jiang, Jie Wu, Guojun Wang, and Huanyang Zheng. 2016. Forming Opinions via Trusted Friends: Time-evolving Rating Prediction Using Fluid Dynamics. *IEEE Transactions on Computers (TC)* 65(4) (2016), 1211–1224.

[20] Aleksandra Karpus, Iacopo Vagliano, and Krzysztof Goczyla. 2017. Serendipitous Recommendations Through Ontology-Based Contextual Pre-filtering. In *BDAS 2017 (Communications in Computer and Information Science)*, Vol. 716. 246–259. https://doi.org/10.1007/978-3-319-58274-0_21

[21] Denis Kotkov, Jari Veijalainen, and Shuaiqiang Wang. 2016. Challenges of Serendipity in Recommender Systems. In *WEBIST 2016*. SciTePress, 251–256. https://doi.org/10.5220/0005879802510256

[22] Denis Kotkov, Jari Veijalainen, and Shuaiqiang Wang. 2018. How does serendipity affect diversity in recommender systems? A serendipity-oriented greedy algorithm. *Computing* (2018), 1–19.

[23] Denis Kotkov, Shuaiqiang Wang, and Jari Veijalainen. 2016. A survey of serendipity in recommender systems. *Knowl.-Based Syst.* 111 (2016), 180–192. https://doi.org/10.1016/j.knosys.2016.08.014

[24] Xueqi Li, Wenjun Jiang, Weiguang Chen, Jie Wu, and Guojun Wang. 2019. HAES: A New Hybrid Approach for Movie Recommendation with Elastic Serendipity. In *CIKM 2019*. ACM, 1503–1512. https://doi.org/10.1145/3357384.3357868

[25] Weizhi Ma, Min Zhang, Yue Cao, Woojeong Jin, Chenyang Wang, Yiqun Liu, Shaoping Ma, and Xiang Ren. 2019. Jointly Learning Explainable Rules for Recommendation with Knowledge Graph. In *WWW 2019*. ACM, 1210–1221. https://doi.org/10.1145/3308558.3313607

[26] Valentina Maccatrozzo, Manon Terstall, Lora Aroyo, and Guus Schreiber. 2017. SIRUP: Serendipity In Recommendations via User Perceptions. In *IUI 2017*. ACM, 35–44. https://doi.org/10.1145/3025171.3025185

[27] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *SIGIR 2015*. ACM, 43–52. https://doi.org/10.1145/2766462.2767755

[28] James McInerney, Benjamin Lacker, Samantha Hansen, Karl Higley, Hugues Bouchard, Alois Gruson, and Rishabh Mehrotra. 2018. Explore, exploit, and explain: personalizing explainable recommendations with bandits. In *RecSys 2018*. ACM, 31–39. https://doi.org/10.1145/3240323.3240354

[29] Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010*. ISCA, 1045–1048. http://www.isca-speech.org/archive/interspeech_2010/i10_1045.html

[30] Xi Niu. 2018. An Adaptive Recommender System for Computational Serendipity. In *ICTIR 2018*. ACM, 215–218. https://doi.org/10.1145/3234944.3234974

[31] Xi Niu, Fakhri Abbas, Mary Lou Maher, and Kazjon Grace. 2018. Surprise Me If You Can: Serendipity in Health Information. In *CHI 2018*. ACM, 23. https://doi.org/10.1145/3173574.3173597

[32] Gaurav Pandey, Denis Kotkov, and Alexander Semenov. 2018. Recommending Serendipitous Items using Transfer Learning. In *CIKM 2018*. ACM, 1771–1774. https://doi.org/10.1145/3269206.3269268

[33] Carl Edward Rasmussen. 1999. The Infinite Gaussian Mixture Model. In *NIPS 1999*. The MIT Press, 554–560. http://papers.nips.cc/paper/1745-the-infinite-gaussian-mixture-model

[34] Richard A Redner and Homer F Walker. 1984. Mixture densities, maximum likelihood and the EM algorithm. *SIAM review* 26, 2 (1984), 195–239.

[35] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. 2017. Dynamic Routing Between Capsules. In *NIPS 2017*. 3859–3869. http://papers.nips.cc/paper/6975-dynamic-routing-between-capsules

[36] Alan Said, Ben Fields, Brijnesh J. Jain, and Sahin Albayrak. 2013. User-centric evaluation of a K-furthest neighbor collaborative filtering recommender algorithm. In *CSCW 2013*. 1399–1408. https://doi.org/10.1145/2441776.2441933

[37] Jiaxi Tang, Francois Belletti, Sagar Jain, Minmin Chen, Alex Beutel, Can Xu, and Ed H. Chi. 2019. Towards Neural Mixture Recommender for Long Range Dependent User Sequences. In *WWW 2019*. 1782–1793. https://doi.org/10.1145/3308558.3313650

[38] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous Graph Attention Network. In *WWW 2019*. ACM, 2022–2032. https://doi.org/10.1145/3308558.3313562

[39] Jiaxuan You, Yichen Wang, Aditya Pal, Pong Eksombatchai, Chuck Rosenberg, and Jure Leskovec. 2019. Hierarchical Temporal Convolutional Networks for Dynamic Recommender Systems. In *WWW 2019*. ACM, 2236–2246. https://doi.org/10.1145/3308558.3313747

[40] Shuai Yu, Yongbo Wang, Min Yang, Baocheng Li, Qiang Qu, and Jialie Shen. 2019. NAIRS: A Neural Attentive Interpretable Recommendation System. In *WSDM 2019*. ACM, 790–793. https://doi.org/10.1145/3289600.3290609

[41] Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. 2019. ProNE: Fast and Scalable Network Representation Learning. In *IJCAI 2019*. ijcai.org, 4278–4284. https://doi.org/10.24963/ijcai.2019/594

[42] Yuan Cao Zhang, Diarmuid Ó Séaghdha, Daniele Quercia, and Tamas Jambor. 2012. Auralist: introducing serendipity into music recommendation. In *WSDM 2012*. 13–22. https://doi.org/10.1145/2124295.2124300