# Task-optimized User Clustering based on Mobile App Usage for Cold-start Recommendations

Bulou Liu*
Tencent Security Big Data Lab
Beijing, China
blowliu@tencent.com

Bing Bai*
Tencent Security Big Data Lab
Beijing, China
icebai@tencent.com

Weibang Xie
Tencent Inc.
Guangzhou, China
colinxie@tencent.com

Yiwen Guo
Independent Researcher
Beijing, China
guoyiwen89@gmail.com

Hao Chen
University of California, Davis
California, USA
chen@ucdavis.edu

## ABSTRACT

This paper reports our recent practice of recommending articles to cold-start users at Tencent. Transferring knowledge from information-rich domains to help user modeling is an effective way to address the user-side cold-start problem. Our previous work demonstrated that general-purpose user embeddings based on mobile app usage helped article recommendations. However, high-dimensional embeddings are cumbersome for online usage, thus limiting the adoption. On the other hand, user clustering, which partitions users into several groups, can provide a lightweight, online-friendly, and explainable way to help recommendations. Effective user clustering for article recommendations based on mobile app usage faces unique challenges, including (1) the gap between an active user's behavior of mobile app usage and article reading, and (2) the gap between mobile app usage patterns of active and cold-start users. To address the challenges, we propose a tailored Dual Alignment User Clustering (DAUC) model, which applies a sample-wise contrastive alignment to eliminate the gap between active users' mobile app usage and article reading behavior, and a distribution-wise adversarial alignment to eliminate the gap between active users' and cold-start users' app usage behavior. With DAUC, cold-start recommendation-optimized user clustering based on mobile app usage can be achieved. On top of the user clusters, we further build candidate generation strategies, real-time features, and corresponding ranking models without much engineering difficulty. Both online and offline experiments demonstrate the effectiveness of our work.

## CCS CONCEPTS

• **Information systems → Data mining**; *Personalization*; • **Theory of computation → Unsupervised learning and clustering**.

* Equal contributions from both authors. This work was done when Bulou Liu worked as an intern at Tencent.

*KDD '22, August 14–18, 2022, Washington, DC, USA*
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539105

## KEYWORDS

user clustering, mobile app, recommender system, cold-start recommendation

**ACM Reference Format:**
Bulou Liu, Bing Bai, Weibang Xie, Yiwen Guo, and Hao Chen. 2022. Task-optimized User Clustering based on Mobile App Usage for Cold-start Recommendations. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3534678.3539105

## 1 INTRODUCTION

Personalized recommendations have become a critical factor in improving user satisfaction and encouraging more transactions. How to accurately recommend items to users without sufficient feedback, i.e., cold-start users, is a long-standing problem that practitioners often face in real applications. This problem becomes more prominent when new recommendation tabs are launched or for specific scenarios where light users are dominant. In practice, two kinds of methods are widely adopted to address this problem. Firstly, we can capture the preference of cold-start users more efficiently by bandit approaches [10, 14]. Secondly, we can also transfer knowledge from information-rich domains to help user modeling, for example, from the mobile app usage behavior [1, 12, 25]. As a motivating example, a user with the popular game "*Honor of Kings*" installed may be interested in tutorials about how to play better, and a user with the photography editing app "*Snapseed*" installed may be interested in travel journals and articles on how to take better photos.

In our previous work [25], we extracted general-purpose user embeddings based on mobile app usage. We demonstrated that the embeddings helped improve the quality of online article recommendations. High-dimensional general-purpose embeddings manage to retain as much information as possible from mobile app usage behavior. However, they may also have the following disadvantages. Firstly, embeddings are cumbersome for storage, transmission, and processing [4]. This may bring engineering challenges and high expenses in scenarios requiring low latency or having large request volumes, thus limiting the adoption. Secondly, embeddings may be hard to interpret, which brings extra difficulties for verification and debugging [5]. On the other side, user clustering, which groups users into a certain number of clusters, can provide a lightweight, online-friendly, explainable way to complement high-dimensional

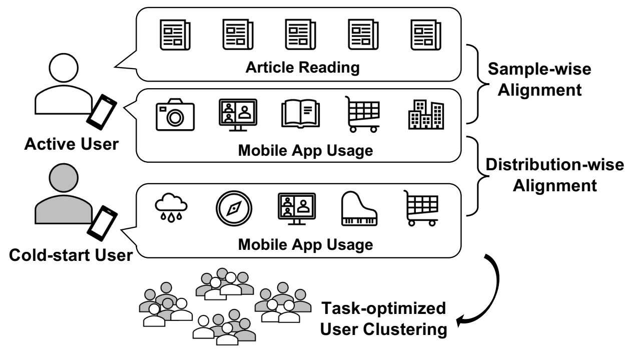


**Figure 1: Illustration of the basic idea of DAUC. By using the active users' mobile app usage behavior as a bridge, we can transfer the knowledge from active users' article reading behavior to cold-start users' mobile app usage behavior, thus improving the performance of recommending articles to cold-start users.**

embeddings [9]. Especially, within each user cluster, the behavioral information of active users can be transferred to cold-start users, thus greatly benefiting cold-start recommendations.

We have been working towards better user clustering based on mobile app usage and utilizing the clusters to improve online article recommendations at Tencent since mid-2021. In this paper, we report our most recent practice. To start with, a naïve idea is to apply clustering methods (e.g., $k$-means) with the app usage features. However, this scheme has two notable shortcomings.

- **The behavior of mobile app usage and article reading may have a significant gap.** As a result, clustering models optimizing the tightness and the separation of mobile app usage behavior may fail to perform well for article reading behavior. This problem is not that serious for user embedding models [25] since high-dimensional embeddings can retain as much helpful information as possible, and downstream models can find effective signals by themselves with the help of supervision. However, clustering is a highly abstractive operation. Eliminating the gap between these two kinds of behavior is essential for recommendation-oriented user clustering.
- **Active and cold-start users' mobile app usage patterns may have significant differences.**[1] Active users tend to be heavy phone users and install more niche apps. Clustering models may spot these domain differences and generate less effective clusters for article recommendations. As an extreme example, models may cluster active and cold-start users into separate groups, with the result that cold-start users will not receive enough transferred knowledge, and the recommendation performance will be poor.

[1]As a quick, illustrative experiment, we train a simple neural network classifier to distinguish active and cold-start users (users with no or less than three clicks in a month) using mobile app usage data. The model achieved an AUC score of 0.676, indicating that domain differences do exist between mobile app usage patterns of active and cold-start users.

To address the problems, we propose a tailored Dual Alignment User Clustering (DAUC) model for cold-start recommendation-optimized user clustering based on mobile app usage data. The overall idea is illustrated in Figure 1. Firstly, to eliminate the gap between active users' mobile app usage and article reading behavior, we regard the two kinds of behavioral information as two views of specific users and use a multi-view contrastive loss to align the *sample-wise* representations. The intuition is to force the representations of mobile app usage behavior and the representations of article reading behavior of a specific user to be close and push the representations of different users away. Secondly, to eliminate the gap between active users' and cold-start users' app usage behavior, we introduce adversarial training to reduce the domain difference in a *distribution-wise* manner. Using active users' mobile app usage behavior as a bridge, we transfer the knowledge from active users' article reading behavior to cold-start users' mobile app usage behavior for better user clustering.

With the help of the proposed DAUC model, we can generate task-optimized, online-friendly, and high-coverage user clusters, based on which we can further design different strategies for candidate generation and build real-time features for ranking without much engineering difficulty. For example, in our practice, we adopt clustering-based candidate generation by selecting items with high click-through rate (CTR) scores or high click counts within different ranges of periods. In addition, for ranking, we construct real-time user cluster portraits by counting the statistical behavior data of user groups, and build real-time article portraits by analyzing the feedback of different user clusters on the articles. These strategies and features are easy to implement, generalize well from active to cold-start users, and contribute a lot to our online recommendations.

The main contribution of this paper is summarized as follows.

- We introduce our recent practice towards task-optimized user clustering based on mobile app usage and how we use the clustering results to improve online recommendations.
- We propose the Dual Alignment User Clustering model tailored for the task. By using active users' app behavior as a bridge, we are able to transfer the knowledge from active users' article reading behavior to clustering cold-start users based on mobile app usage data. The model is scalable for industrial data.
- Extensive online and offline experiments verify the effectiveness of the proposed model, which has been deployed in a real system at Tencent and boosted recommendation performance in daily business.

## 2 SYSTEM OVERVIEW

In this section, we introduce our system from a high-level perspective. We review the background, the data processing procedure, serving and updating workflow, and present how we use the user clusters to improve the recommendations.

### 2.1 Background

*Tencent Mobile Manager* is currently the most prevalent mobile security and management app in China. We provide article recommendations as an auxiliary functionality. For example, users can reach personalized content feeds, including news, articles, and short videos, from the "Good Morning" tab of Tencent Mobile Manager and the "Find" tab of Tencent Wi-Fi Manager, a wingman app of Tencent Mobile Manager.

Since we adopt an active exposure strategy, a significant share of page views and clicks are by less active users or even cold-start users. For example, about 75% of exposed users in our scenario had no or less than three clicks in the previous month, and these users contributed more than 50% of total page views and 25% of total clicks. As a result, improving the recommendation quality for these cold-start users plays an essential role in our scenario.

## 2.2 Data Preprocessing

First, we need to preprocess the raw data to obtain the initial features. We introduce the data preprocessing procedure from two aspects, i.e., app usage behavior data and article reading behavior data.

For app usage behavior data, we select apps based on the following rules. (1) Some top-ranked apps that cannot provide user preference information, like "*WeChat*", are manually excluded. (2) Preinstalled apps by the manufacturers are excluded as they are of high volume but only serve as an indicator of mobile phone brand. (3) One app may have multiple `package_name` on different brands and models of smartphones and some niche apps may offer very similar functionalities, we merge them into one `app_name`. Then we select top apps according to install rates on phones. Finally, we keep about 10 thousand distinct merged apps for consideration. Note that different from [25] which considers retention, installation, and uninstallation collectively, in this work, we only consider the retention information, i.e., what apps are currently installed on the phones. This is due to a trade-off between performance and workload, as obtaining installation and uninstallation information requires scanning about half a year of records which is a cumbersome operation. Finally, the mobile app usage data is represented by multi-hot binary vectors where "1" represents that the corresponding app is currently installed on the phone.

For article reading behavior data, we first select users with more than three clicks in one month as active users and filter spam users with a threshold, then we calculate the averaged article embeddings of clicked and unclicked articles respectively and use their concatenation to represent users' article reading preference. The article embeddings are 128-dimensional pretrained dense vectors generated with the title, the body, the category, the tags, the cover pictures, etc. Note that the article reading behavior data is only necessary for model training.

## 2.3 Serving and Updating

For serving, we generate user clusters with the trained model and the app usage behavior data. Then we upload the results to the data center based on DCache[2], a distributed in-memory NoSQL system.

To capture the latest preferences of users, we need to update the predictions regularly. We mainly use feature updating [25], i.e., keeping the model fixed and only updating the features of users. With this strategy, we take into consideration the up-to-date behavior of users and can hold a consistent semantic clustering space. The model is only updated when significant offline improvements are achieved.

Another challenge is the large volume of cold-start users, as all users with the app installed are potential users of the recommendation tab. To strike a balance between computational burden and feature timeliness, we use a hierarchical update strategy. Every day, we update the clustering results of users with exposure the day before, and these users can cover most of the query traffic; every month, we update the clusters of all users with our app installed. This strategy helps to minimize computational cost and guarantee coverage and timeliness.

## 2.4 Applications with User Clusters

The user clusters are optimized for article reading behavior and cover almost all users. It can be valuable for online recommendations, especially to cold-start users. We utilize the clusters in both the candidate generation and the ranking stages.

*Candidate generation.* Clustering is much more online-friendly compared with high-dimensional dense embeddings. We can easily build real-time candidate generation strategies by analyzing the behavior of user groups, i.e., user-to-group-to-item (U2G2I) recommendations. For example, within a user group, items with high CTR in a relatively short period may be emerging hot news, and items with high click count in a long period may be classical articles. After filtering out duplicates and exposed ones, we recall top-ranked items based on their CTR scores and clicks over one hour, one day, and seven days as a part of the multi-way candidate generation systems.

*Ranking.* Our ranking system is based on a multi-task neural network model, i.e., Multi-gate Mixture-of-Experts (MMoE) [26]. We estimate the (weighted) expected reading time after exposure for ranking. We build a set of numerical features for the ranking model with the help of user clusters. On the user side, the user group portrait can act as a supplementary to the user portrait. Many feature engineering strategies for users can also be applied to user clusters. For example, we calculate statistics like click counts, CTR scores, and reading time distribution over different categories and tags of articles in one day, seven days, and one month. These features can provide valuable information for cold-start users that do not give enough feedback regarding their personal preferences. On the item side, how different groups of users rate the item also indicates the property of the content. Similar statistics can also be calculated from the item view. We mainly use a coverage threshold for feature selection to filter infrequent features. We also add precrossed features, i.e., the statistics of the exact user group over the exact item (item category/tag). Although deep neural networks are supposed to learn feature combination patterns from ID features automatically [17], our experiments show that these handcrafted features can still contribute to overall performance, especially for cold-start recommendations.

## 3 DUAL ALIGNMENT USER CLUSTERING

This section defines the notations of user behavior, followed by the detailed structure of the proposed network. Then, we elaborate on our designs for eliminating the gap between active users' mobile

[2]https://github.com/Tencent/DCache

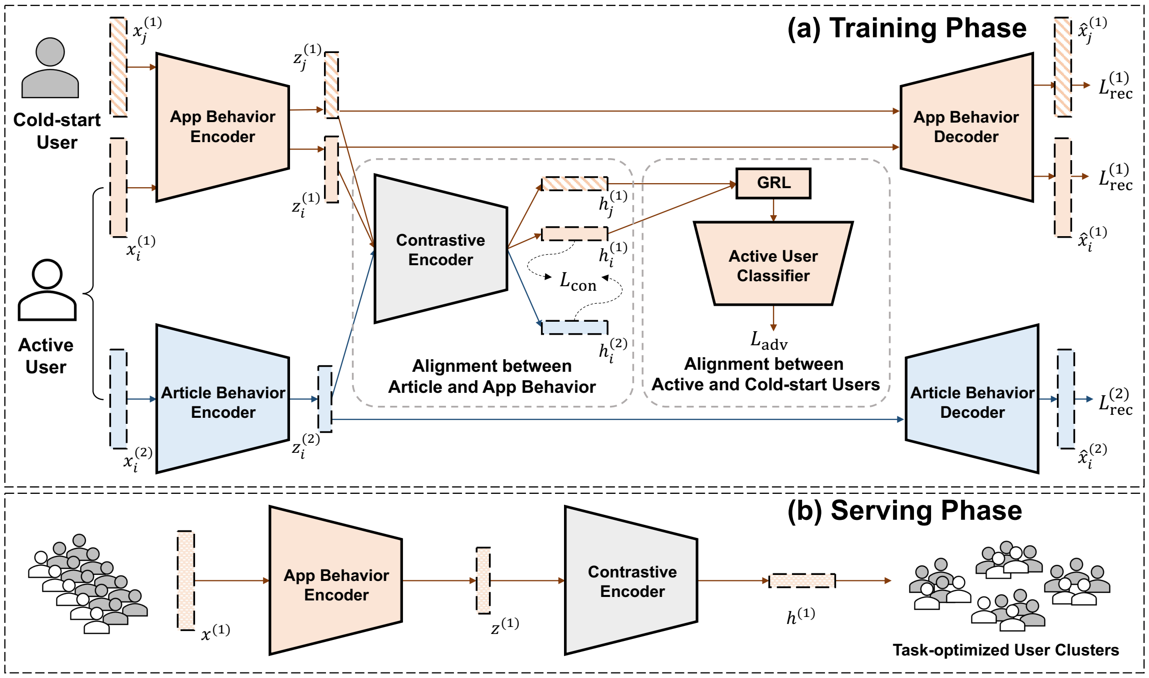


**Figure 2: The overview of the proposed DAUC model. The model is tailored for cold-start recommendation-optimized user clustering based on mobile app usage data. In the training phase, the app behavior autoencoder and article behavior autoencoder aim to learn good representations for users' app usage behavior and article reading behavior, respectively. Alignment between news and app behavior aims to provide a *sample-wise* approach of aligning representations from the two views, and alignment between active and cold-start users aims to provide a *distribution-wise* approach of aligning representations from the two kinds of users. In the serving phase, the *k*-means algorithm is employed to generate recommendation-optimized user clusters based on the aligned representations of app usage behavior.**

app usage and article reading behavior, and the gap between active and cold-start users' app usage behavior. Finally, we present the multi-objective joint training scheme for model optimization.

## 3.1 Notations

As stated in Section 2.2, the behavior of each user are preprocessed into two kinds of data, i.e., app usage behavior data and article reading behavior data. $U$ is the number of users and $U_a$ is the number of active users.

**App usage behavior.** The app usage behavior of user $i$ can be represented by a multi-hot vector $x_i^{(1)} = (x_{i1}^{(1)}, \ldots, x_{im}^{(1)}, \ldots, x_{iM}^{(1)})$ where $M$ is the number of considered apps. $x_{im}^{(1)} = 1$ when app $m$ is installed by user $i$ and $x_{im}^{(1)} = 0$ otherwise.

**Article reading behavior.** The article reading behavior of user $i$ can be represented by a dense vector $x_i^{(2)} \in \mathbb{R}^N$, which is the concatenation of the averaged pre-trained embeddings of clicked and unclicked articles.

Note that in the rest of the paper, we may omit the subscript $i$ that indicates the user when there is no confusion.

## 3.2 Model Overview

As shown in Figure 2, the proposed model DAUC contains two phases: the training phase and the serving phase. Details about the network structure are as follows.

*Training Phase.* The two kinds of user behavior data tend to contain consensus and complementary information, and how to extract consensus representations is the major challenge in our task. We first apply an app behavior autoencoder [8] and an article behavior autoencoder to learn representations for users' app usage behavior and article reading behavior, respectively. Note that the article behavior autoencoder just takes the active users' reading behavior as input. The reconstruction from input to output not only helps each view keep the view-specific information, but also makes subsequent similarity computing on latent features more reliable and prevents model collapse. Specifically, considering the $v$-th view $x^{(v)}$, the corresponding encoder and decoder are represented as follows:

$$\begin{aligned} z^{(v)} &= E^{(v)}\left(x^{(v)}; \theta_e^{(v)}\right) \\ \widehat{x}^{(v)} &= D^{(v)}\left(z^{(v)}; \theta_d^{(v)}\right), \end{aligned} \tag{1}$$

where $v \in \{1,2\}$, $\theta_e^{(v)}$ and $\theta_d^{(v)}$ denote the autoencoder network parameters, $z^{(v)} \in \mathbb{R}^{D_{ae}}$ is learned $D_{ae}$-dimensional latent representations and $\widehat{x}^{(v)} \in \mathbb{R}^{D_v}$ is reconstructed output.

Then we utilize a contrastive encoder to map the latent representations to aligned representations as follows:

$$h^{(v)} = f_{\text{con}}\left(z^{(v)}; \theta_c\right), \tag{2}$$

where $\theta_c$ denotes the parameters of contrastive encoder $f_{\text{con}}(\cdot)$, and $h^{(v)} \in \mathbb{R}^{D_h}$. To further eliminate the gap between active users' mobile app usage and article reading behavior, we regard the two kinds of behavior information as two views of specific users and use a multi-view contrastive loss to align the *sample-wise* representations. More details are provided in Section 3.3. And to eliminate the gap between active users' and cold-start users' app usage behavior, we introduce adversarial training to reduce the domain difference between active users and cold-start users in a *distribution-wise* manner. More details are provided in Section 3.4.

*Serving Phase.* After the training phase, we can obtain aligned representations $h^{(1)}$, which is generated from mobile app usage behavior, and optimized for article reading behavior. Following the traditional fashion [11, 13, 21, 24], we utilize the $k$-means algorithm [16] to generate user clusters based on the aligned representations of app usage behavior. Our experiments show that this scheme works well.

## 3.3 Alignment between News and App Behavior

To optimize the clusters for news reading behavior, we need to force the model to concentrate on the helpful information from app usage data and overlook the irrelevant part. Inspired by recent works with contrastive learning [23], we propose to eliminate the gap in a contrastive way, which is achieved by a multi-view contrastive loss defined as

$$\begin{aligned}\mathcal{L}_{\text{con}} = -\frac{1}{2U_a}\sum_{i=1}^{U_a}\Bigg(&\log\frac{e^{d_{i,i}^{(1)(2)}/\tau}}{\sum_{j=1,j\neq i}^{U_a} e^{d_{i,j}^{(1)(1)}/\tau}+\sum_{j=1}^{U_a} e^{d_{i,j}^{(1)(2)}/\tau}}\\ &+\log\frac{e^{d_{i,i}^{(2)(1)}/\tau}}{\sum_{j=1,j\neq i}^{U_a} e^{d_{i,j}^{(2)(2)}/\tau}+\sum_{j=1}^{U_a} e^{d_{i,j}^{(2)(1)}/\tau}}\Bigg),\end{aligned} \tag{3}$$

where $\tau$ is the temperature hyperparameter for multi-view contrastive loss, and $d_{i,j}^{(1)(2)}$ compute the cosine similarity of two representations $h_i^{(1)}$ of user $i$ and $h_j^{(2)}$ of user $j$, i.e., $d_{i,j}^{(1)(2)} = \frac{h_i^{(1)\mathrm{T}} h_j^{(2)}}{\left\|h_i^{(1)}\right\|\cdot\left\|h_j^{(2)}\right\|}$.

The multi-view contrastive loss in Equation (3) forces the hidden representations of news and app behavior from the same user, i.e., $(h_i^{(1)}, h_i^{(2)})$, to be closer than the representations from different users, i.e., $(h_i^{(1)}, h_j^{(2)})$, $(h_i^{(1)}, h_j^{(1)})$, and $(h_i^{(2)}, h_j^{(2)})$, thus can align the different kinds of behavior information [20] and eliminate the gap. Note that the loss in Equation (3) can be efficiently optimized by mini-batch training and negative sampling.

## 3.4 Alignment between Active and Cold-start Users

As discussed before, active and cold-start users' mobile app usage patterns may also have significant differences. Clustering models may spot these domain differences and generate less effective clusters for cold-start recommendations. Therefore, the aligned representations $h^{(1)}$ are expected to be domain-independent. Namely, $h^{(1)}$ should only contain the shared information of active and cold-start users, and omit the information about whether a user is active or not. Hence, we introduce an adversarial learning mechanism to ensure that no domain-specific information can be memorized during the training stage [6, 19]. Specifically, we introduce an active user classifier over $h^{(1)}$ to calculate the probability that $h^{(1)}$ belongs to active or cold-start users during the training:

$$p_{\text{active}}(\cdot|h^{(1)}) = \text{softmax}\left(f_{\text{adv}}(\cdot|h^{(1)};\phi)\right), \tag{4}$$

where $\phi$ denotes the parameters of the active user classifier $f_{\text{adv}}(\cdot)$. We optimize $\phi$ by minimizing the negative log-likelihood, such that the domain classifier can classify $h^{(1)}$ to its true domain:

$$\mathcal{L}_{\text{adv}} = -\frac{1}{U}\sum_{i=1}^{U}\log\left(p_{\text{active}}(y_i|h_i^{(1)})\right), \tag{5}$$

and $\{(x_i^{(1)}, y_i)\}$ is the training set such that user $i$ belongs to domain $y_i$, i.e., active or cold-start users.

In this sense, parameters $\phi$ are adjusted to be more sensitive on the domain-specific information encoded by $h^{(1)}$. Because we expect the aligned representations $h^{(1)}$ carry no domain-specific information, such that the classifier can not perform the category classification precisely. It means that $\theta = \{\theta_e^{(1)}, \theta_c\}$ is optimized to make $h^{(1)}$ indistinguishable by the classifier. To achieve the reverse purpose, we add the Gradient Reversal Layer (GRL) [6] between $h^{(1)}$ and the domain classifier. We can consider GRL as a pseudo-function $R_\lambda(\chi)$:

$$R_\lambda(\chi) = \chi;\ \frac{\partial R_\lambda}{\partial \chi} = -\lambda I, \tag{6}$$

where $\lambda$ controls the importance of adversarial learning. This means that the gradient for parameters $\theta$ are reversed and scaled by a parameter $\lambda$, which is equivalent to reducing the category-specific information in $h^{(1)}$. In a word, with the help of the Gradient Reversal Layer, the network involving $\theta$ and the adversarial classifier play a minimax game to optimize $\theta$ and $\phi$, thus the distribution-wise alignment of $h^{(1)}$ between active and cold-start users is achieved.

## 3.5 Model Training Scheme

To obtain task-optimized representations for clustering, DAUC jointly optimizes three objects by integrating the autoencoder reconstruction loss, multi-view contrastive loss, and adversarial training loss. The overall objective is defined as:

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{con}} + \mathcal{L}_{\text{adv}}, \tag{7}$$

where the second term $\mathcal{L}_{\text{con}}$ and the third term $\mathcal{L}_{\text{adv}}$ are calculated with Equation (3) and (5), respectively. For the first term $\mathcal{L}_{\text{rec}}$, considering that the app usage behavior is represented by multi-hot vectors, we choose the sigmoid cross-entropy as the reconstruction loss function:

$$\mathcal{L}_{\text{rec}}^{(1)} = -\frac{1}{UM}\sum_{i=1}^{U}\sum_{m=1}^{M}\left(x_{im}^{(1)}\log\left(\sigma(\widehat{x}_{im}^{(1)})\right) + (1 - x_{im}^{(1)})\log\left(1-\sigma(\widehat{x}_{im}^{(1)})\right)\right), \tag{8}$$

where $\sigma(\cdot)$ is the sigmoid function, and $M$ is the number of considered apps. Since the article reading behavior is represented by dense vectors, the reconstruction loss of article reading behavior data is measured by Mean Squared Error (MSE):

$$\mathcal{L}_{\text{rec}}^{(2)} = \frac{1}{U_a}\sum_{i=1}^{U_a}\left\|x_i^{(2)} - \widehat{x}_i^{(2)}\right\|_2^2. \tag{9}$$

The total reconstruction loss is calculated as follows:

$$\mathcal{L}_{\text{rec}} = \mathcal{L}_{\text{rec}}^{(1)} + \mathcal{L}_{\text{rec}}^{(2)}. \tag{10}$$

**Table 1: The statistics of experimental datasets.**

| | Industrial | Taobao |
|---|---|---|
| # of users | 4,921,532 | 37,446 |
| # of active users | 2,391,298 | 16,647 |
| # of items | 428,353 | 289,175 |
| # of impressions | 437,705,556 | 4,143,735 |
| # of clicks | 36,981,413 | 137,558 |
| *avg.* # of installed apps | 30.7 | 116.1 |

The overall loss of Equation (7) can be efficiently optimized with mini-batch training. We roughly experimented with different weights on the loss terms and found that equal weights for all terms worked well.

## 4 OFFLINE EXPERIMENTS

This section reports the offline experimental results. We first introduce the dataset and baselines, then present the evaluation scheme. Offline evaluation results and analyses are presented at last.

### 4.1 Dataset

We evaluated the proposed model with the industrial dataset from our article recommendation service and a proxy dataset built with the Taobao Ad Display/Click Data.

*Industrial dataset.* This dataset was collected from the log of our online article recommendations and covered a period of approximately 50 days. We filter potential spam users with more than 5,000 clicks in the period. We chose the first 70% impression data sorted by time as the training set, the next 10% as the validation set, and the last 20% as the test set. Detailed statistics are in Table 1.

*Taobao dataset.* To test the generalization of our model, we evaluated the performance on publicly-available datasets. Although there were many datasets for evaluating recommender systems, unfortunately, we found no datasets that could perfectly match the property of our scenario. So we built a proxy dataset with the Taobao Ad Display/Click Dataset[3], as this dataset provided display data both with and without clicks. To simulate the cross-domain transferring property of our task, we used the click behavior on cheap items and expensive items as proxies. Specifically, clicks on items cheaper than or equal to 160RMB were used as the proxy mobile app usage behavior. Clicks on items more expensive than 160RMB were used as the proxy article reading behavior. We also chose the first 70% impression data sorted by time as the training set, the next 10% as the validation set, and the last 20% as the test set. The statistics of this proxy dataset is presented in Table 1, and more details about dataset preprocessing is reported in Appendix A.1.[4]

### 4.2 Baselines

For a comprehensive evaluation, we compared our method with the following baselines.

- **Random**. This method splits the uses into clusters randomly.
- **Profile**. This is the method serving the online traffic before DAUC. It splits users according to the cartesian product of user profile (`gender`, `age`, and `city_level`), and outputs nearly 200 user clusters.
- ***k*-means** [16]. This is one of the most classical clustering methods. We use the mini-batch variant [16] to tackle the large-scale data.
- **DEC** [22]. This method learns a deep neural network-based mapping from the data space to a lower-dimensional feature space in which it iteratively optimizes a clustering objective.
- **IDEC** [7]. This method improves DEC with local structure preservation.
- **DCCAE** [21]. This method optimizes both canonical correlation analysis (CCA) and reconstruction-based objectives. It achieved the best results on most tasks by Wang et al. [21] among many variants of CCA-based methods.
- **COMPLETER** [11]. This method was proposed recently as a state-of-the-art model for multi-view clustering.

Among the baseline methods, Random and Profile do not use mobile app usage behavior, *k*-means, DEC and IDEC only consider the mobile app usage behavior while ignoring the gap between two different kinds of user preference, and DCCAE and COMPLETER use both behavior information for user clustering. Note that for the multi-view baselines, i.e., DCCAE and COMPLETER, we only use the representations from the mobile app view for clustering, as cold-start users do not provide sufficient article reading behavior.

### 4.3 Evaluation Scheme

We report the offline evaluation scheme in this section, including the evaluation protocol, the metrics, and how the hyper-parameters were set.

*Evaluation protocol.* We used an offline replay method to evaluate the clustering results. We first trained the models using the training set, with which we clustered the users into a fixed number of user groups. The evaluation worked in the following way. We chronologically divided the testing set into ten subsets, and for each subset, we calculated the CTR scores of articles within a slicing window before the subset, and sorted the items according to CTR scores in a decent manner. For every user in the test subset, we recommend items according to CTR scores after filtering the already-exposed items by the user. This protocol simulated a basic CTR-based U2G2I strategy commonly used in industrial recommender systems. We evaluate both the overall recommendation performance and the recommendation performance for cold-start users only.

*Metrics.* We used two commonly used metrics for evaluating recommender systems, i.e., Recall@$K$ and Normalized Discounted Cumulative Gain@$K$ (NDCG@$K$). Both metrics are the higher, the better. Recall@$K$ treat the top $K$ selected items equally, while NDCG@$K$ pay more attention to the very top of recommendation list. Thus by evaluating with both metrics, we can have a better understanding of the recommendation performance of each methods. During our evaluations, we set $K$ to 50 for the Industrial dataset, and $K$ to 20 for the Taobao dataset. We report the averaged results over five runs for all the methods and apply the student t-test to

[3]https://tianchi.aliyun.com/dataset/dataDetail?dataId=56&lang=en-us

[4]We note that the Taobao dataset comes from an advertising scenario, which, unlike recommendation scenarios, may be significantly affected by advertisers' behavior.

**Table 2: Offline evaluation results for user clustering methods. Boldface denotes the highest score, underline indicates the best result of the baselines, and *Improv.* present the relative improvement of DAUC over the best result of the baselines. $*/**$ denotes that DAUC performs significantly better than the baseline at $p < 0.05/0.01$ level with a two-tailed pairwise t-test.**

| Model | Industrial (overall) | | Industrial (cold-start) | | Taobao (overall) | | Taobao (cold-start) | |
|---|---|---|---|---|---|---|---|---|
| | Recall@50 | NDCG@50 | Recall@50 | NDCG@50 | Recall@20 | NDCG@20 | Recall@20 | NDCG@20 |
| Random | $0.0149^{**}$ | $0.0047^{**}$ | $0.0084^{**}$ | $0.0032^{**}$ | $0.1421^{**}$ | $0.0679^{**}$ | $0.2095^{**}$ | $0.1083^{**}$ |
| Profile | $0.0286^{**}$ | $0.0097^{**}$ | $0.0253^{**}$ | $0.0092^{**}$ | – – | – – | – – | – – |
| $k$-means | $0.4850^{**}$ | $0.1578^{**}$ | $0.5849^{**}$ | $0.1886^{**}$ | $0.1962^{**}$ | $0.0878^{**}$ | $0.2923^{**}$ | $0.1376^{**}$ |
| DEC | $0.4308^{**}$ | $0.1314^{**}$ | $0.5090^{**}$ | $0.1532^{**}$ | $0.1571^{**}$ | $0.0751^{**}$ | $0.2015^{**}$ | $0.1142^{**}$ |
| IDEC | $0.4981^{**}$ | $0.1590^{**}$ | $0.5366^{**}$ | $0.1693^{**}$ | $0.1448^{**}$ | $0.0724^{**}$ | $0.2642^{**}$ | $0.1471^{*}$ |
| DCCAE | $0.5230^{**}$ | $0.1706^{**}$ | $0.6257^{**}$ | $0.2044^{**}$ | $0.2169^{**}$ | $0.1068^{**}$ | $0.2942^{**}$ | $0.1455^{*}$ |
| COMPLETER | $\underline{0.5384}^{*}$ | $\underline{0.1913}^{*}$ | $\underline{0.6436}^{*}$ | $\underline{0.2297}^{**}$ | $\underline{0.2373}^{**}$ | $\underline{0.1200}^{**}$ | $\underline{0.3093}^{**}$ | $\underline{0.1476}^{*}$ |
| DAUC | **0.5770** | **0.2181** | **0.6745** | **0.2548** | **0.2813** | **0.1378** | **0.3253** | **0.1556** |
| *Improv.* | +7.17% | +14.00% | +4.80% | +10.93% | +18.54% | +14.83% | +5.17% | +5.42% |

conduct the statistical significance. The formulation of the two metrics are in Appendix A.2.

*Hyper-parameter settings.* The two behavior autoencoder are designed by fully connected layers. The app behavior encoder $E^{(1)}(\cdot)$ is with the dimensions of $M$-800-600-400 and the article behavior encoder $E^{(2)}(\cdot)$ is with the dimensions of $N$-256-400 for both the datasets. The dimensions of the decoder and the corresponding encoder are in reverse order. In terms of the contrastive encoder and the active user classifier, $f_{\text{con}}(\cdot)$ and $f_{\text{adv}}(\cdot)$ are fully connected layers with 400-400-400 dimensions and 400-2 dimensions, respectively for both the datasets. We generate 100 clusters with each method for the Industrial dataset (excluding Profile) and 20 clusters for the Taobao dataset for evaluations. More detailed settings are presented in Appendix A.3.

## 4.4 Evaluation Results

Table 2 shows the results in the two datasets. We can draw the following conclusions from the results, and more analysis is in Appendix B.

**The application of mobile app usage behavior benefits the quality of article recommendations.** We can observe that clustering models which consider the mobile app usage behavior significantly outperformed Random and Profile in the Industrial dataset. In addition, we also experimented with concatenating the profile information and app usage information together but gained no improvements. Our hypothesis is that (1) the profile data are noisy and (2) the profile information is well entailed in the app usage behavior.

**Eliminating the gap between app usage behavior and article reading behavior helps generate better user clusters for article recommendations.** We can find that the multi-view clustering baselines, i.e., DCCAE and COMPLETER, achieved better recommendation performance than $k$-means, DEC, and IDEC in the Industrial dataset.

**DAUC significantly performs better than all the baselines in the Industrial dataset.** Thanks to the dual alignment strategy, DAUC improved over the best baselines at a significant margin under all metrics.

**DAUC has a good generalization ability to cluster users for cold-start recommendation tasks.** We found that the above qualitative conclusions hold on the Taobao dataset. These results show that the proxy dataset and Industrial dataset have similar task characteristics. And the two kinds of alignment can help generate high-quality user cluster, thus DAUC has a good generalization ability for different recommendation scenarios.

**Table 3: Ablation study on DAUC. $*/**$ denotes that DAUC performs significantly better than the ablation at $p < 0.05/0.01$ level with a two-tailed pairwise t-test.**

| Model | Industrial (overall) | | Industrial (cold-start) | |
|---|---|---|---|---|
| | Recall@50 | NDCG@50 | Recall@50 | NDCG@50 |
| DAUC | **0.5770** | **0.2181** | **0.6745** | **0.2548** |
| $\text{DAUC}_{\neg\text{rec}}$ | $0.5475^{*}$ | $0.1971^{*}$ | $0.6555^{*}$ | $0.2393^{*}$ |
| $\text{DAUC}_{\neg\text{con}}$ | $0.5348^{**}$ | $0.1910^{**}$ | $0.6391^{**}$ | $0.2266^{**}$ |
| $\text{DAUC}_{\neg\text{adv}}$ | $0.5385^{*}$ | $0.1935^{*}$ | $0.6453^{**}$ | $0.2335^{**}$ |

## 4.5 Ablation Study

We conduct an ablation study to evaluate the effects of the components of DAUC. To be specific, we refer to the variation without autoencoders for the reconstruction of multiple views as $\text{DAUC}_{\neg\text{rec}}$, the variation without alignment between news and app behavior as $\text{DAUC}_{\neg\text{con}}$, and the variation without alignment between active and cold-start users as $\text{DAUC}_{\neg\text{adv}}$. Table 3 shows the experimental results on the Industrial dataset. The results indicate that all the proposed components contributed to the final recommendation performance. Specifically, alignment between news and app behavior contributed most significantly to the performance improvement. It indicates that the contrastive learning module effectively transfers knowledge from mobile app usage behavior to article recommendation. Without alignment between active and cold-start users, DAUC experiences a more significant performance degradation in cold-start recommendations. In addition, we present the percentage

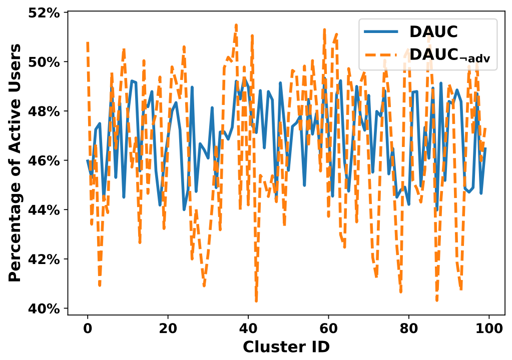
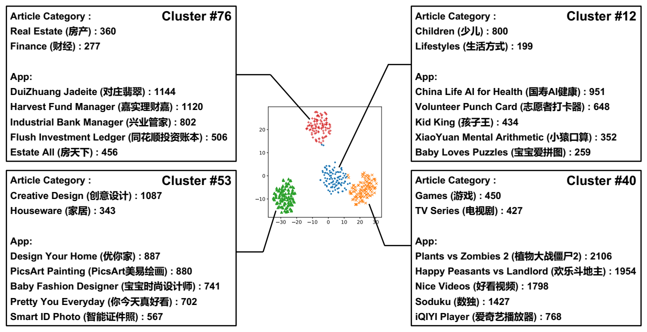


**Figure 3: The percentage of active users in each user cluster by DAUC and DAUC$_{\neg\text{adv}}$. We can find that without the adversarial alignment loss between active and cold-start users, the distribution of active users in user clusters gets more imbalanced.**



Article Category : Cluster #76
Real Estate (房产) : 360
Finance (财经) : 277

App:
DuiZhuang Jadeite (对庄翡翠) : 1144
Harvest Fund Manager (嘉实理财嘉) : 1120
Industrial Bank Manager (兴业管家) : 802
Flush Investment Ledger (同花顺投资账本) : 506
Estate All (房天下) : 456

Article Category : Cluster #12
Children (少儿) : 800
Lifestyles (生活方式) : 199

App:
China Life AI for Health (国寿AI健康) : 951
Volunteer Punch Card (志愿者打卡器) : 648
Kid King (孩子王) : 434
XiaoYuan Mental Arithmetic (小猿口算) : 352
Baby Loves Puzzles (宝宝爱拼图) : 259

Article Category : Cluster #53
Creative Design (创意设计) : 1087
Houseware (家居) : 343

App:
Design Your Home (优你家) : 887
PicsArt Painting (PicsArt美易绘画) : 880
Baby Fashion Designer (宝宝时尚设计师) : 741
Pretty You Everyday (你今天真好看) : 702
Smart ID Photo (智能证件照) : 567

Article Category : Cluster #40
Games (游戏) : 450
TV Series (电视剧) : 427

App:
Plants vs Zombies 2 (植物大战僵尸2) : 2106
Happy Peasants vs Landlord (欢乐斗地主) : 1954
Nice Videos (好看视频) : 1798
Soduku (数独) : 1427
iQIYI Player (爱奇艺播放器) : 768

**Figure 4: A case study of the generated user clusters. We chose four typical user clusters and visualize them in the figure. Then we present the top two article categories according to the TGI of CTR and five corresponding apps with high TGI of installation rates.**

of active users in each cluster on the Industrial dataset by DAUC and DAUC$_{\neg\text{adv}}$ in Figure 3. We can observe that without the adversarial alignment loss between active and cold-start users, the distribution of active users in user clusters gets more imbalanced. These illustrate that the adversarial learning module reduces the domain difference between active and cold-start users and thus may transfer more knowledge to cold-start users.

## 4.6 Case Study

To demonstrate the interpretability of user clusters, we present a case study in Figure 4, where four user clusters are visualized, and the statistics about article reading and app installation behavior are presented. Specifically, we calculated each user cluster's Target Group Index (TGI)[5] of the click-through rates of article categories and the installation rates of apps. We found that the generated clusters built a strong and interpretable connection between app installation and article reading behavior. For example, the CTR of users in Cluster #40 for *Game* related articles was 450% of the CTR of all users, meanwhile, when looking at the apps they install, we did find corresponding evidence, i.e., the users within the group installed a lot of phone gaming apps, including *Plants vs Zombies 2*, *Happy Peasants vs Landlord*, *Soduku*, etc., and the installation rate of these games was over ten times of that of all population.

[5]TGI measures how a target group differs from the total population. For example, TGI = 200 means that the statistic on the target group is 200% of the statistic on the entire population, and TGI = 100 means that the target group is identical to the total population w.r.t. the statistic.

**Table 4: Online improvements of $k$-means and DAUC over Profile.**

| *Improv.* | Model | UV CTR | PV CTR | Engagement | Click |
|---|---|---|---|---|---|
| Overall | $k$-means | +5.04% | +4.85% | +9.71% | +10.35% |
| | DAUC | +9.91% | +8.25% | +13.28% | +16.25% |
| Cold-start | $k$-means | +6.29% | +4.28% | +10.47% | +11.21% |
| | DAUC | +13.70% | +10.46% | +17.55% | +18.87% |

## 5 ONLINE EXPERIMENTS

To verify the performance of the proposed DAUC model and corresponding recommendation strategies, we conducted an online article recommendation A/B test from 2022-01-21 to 2022-01-31. We randomly split online traffic by userIDs with a hashing function. As discussed in Section 2.4, testing different user clustering methods requires a lot of supporting work (a new set of candidate generation modules and real-time features, as well as a new ranking model). We were only able to compare DAUC with the existing online baseline, i.e., Profile, and $k$-means. We considered the following online metrics. UV CTR measures the click-through rate in terms of the user view, and PV CTR measures the click-through rate in terms of the page view. Engagement measures the average reading time of each user with exposure. Click measures the average number of articles each user with exposure clicks.

Table 4 reports the online relative improvement of $k$-means and DAUC over Profile. We can find that bringing the information of app usage behavior is beneficial to article recommendations, and DAUC enjoys more improvement compared with $k$-means. Specifically, we find that the online relative improvements on cold-start users are larger than the overall improvements, which is opposite from the offline evaluation results. This may be due to the limitation of the offline replay evaluation scheme, i.e., even if the algorithm does capture more accurate user preferences and recommend better items, it may not be counted by the metrics when the corresponding items did not get exposed and clicked.

## 6 RELATED WORK

We summarize related work in this section. We mainly review two research fields, i.e., applications with app usage data and multi-view clustering.

## 6.1 Applications with App Usage Data

The boom in smartphones and mobile apps has dramatically changed people's lives. Everyone can customize the functionality of their phones by installing different mobile apps. At the same time, understanding the mobile app usage pattern becomes an effective way to model user preferences for many applications. For example, Bian et al. [1] proposed a macro-micro fusion network for user representation learning on mobile apps by considering both user-app interactions and user-item interactions on some specific app. Liu et al. [12] proposed to make cold-start news recommendations by transferring learning from the app domain to the news domain. Tian et al. [18] proposed a method for the automated segmentation of users' app usage logs into task units. Ochiai et al. [15] proposed to exploit graph convolutional networks for representation learning of mobile app usage.

Our previous work [25] proposed to extract general-purpose user embeddings from mobile app usage and showed that these embeddings were beneficial to a variety of downstream tasks. In contrast, this paper proposes to generate lightweight task-optimized user clusters for cold-start article recommendations with app usage data.

### 6.2 Multi-view Clustering

With advances of data acquisition in real-world applications, data of an underlying signal is often collected from heterogeneous sources or feature subsets. Therefore, multi-view learning has become more and more critical [21]. Multi-view data may contain complementary and consensus information, with which we can suppress view-specific noise, thus improving unsupervised learning like clustering. Early works used matrix factorization [2], spectral clustering [2], and canonical correlation analysis (CCA)-based clustering [3]. Recently, deep neural networks have been introduced to improve multi-view clustering. For example, Wang et al. [21] analyzed several variants based on prior work and found deep canonically correlated autoencoders (DCCAE) worked well. Lin et al. [11] provided a theoretical framework that unified the consistent representation learning and cross-view data recovery.

In our task, we consider the mobile app usage and article reading behavior as two views. Specifically, we aim to improve the clustering results for cold-start users with absent article reading features. Besides, we also consider the domain difference between active users and cold-start users.

## 7 CONCLUSIONS

This paper presents our recent practice for task-optimized user clustering based on mobile app usage. The task is to group users into article-reading behavior-optimized groups so that various online-friendly strategies can be applied to improve the recommendations for cold-start uses. We propose a dual alignment user clustering model to improve the quality of user clusters, by sample-wise contrastive alignment between news and app behavior information and distribution-wise adversarial alignment between active and cold-start users. Offline experiments and ablation studies demonstrate the effectiveness of the proposed method. We further introduce how the user clustering results are used online and show the advantage of the proposed model by online A/B test. Our practice may inspire practitioners about how to utilize knowledge transfer to promote cold-start recommendations.

## REFERENCES

[1] Shuqing Bian, Wayne Xin Zhao, Kun Zhou, Xu Chen, Jing Cai, Yancheng He, Xingji Luo, and Ji-Rong Wen. 2021. A Novel Macro-Micro Fusion Network for User Representation Learning on Mobile Apps. In *Proceedings of 30th The Web Conference.* 3199–3209.

[2] Xiao Cai, Feiping Nie, and Heng Huang. 2013. Multi-view K-means Clustering on Big Data. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence.* 2598–2604.

[3] Kamalika Chaudhuri, Sham M Kakade, Karen Livescu, and Karthik Sridharan. 2009. Multi-view clustering via canonical correlation analysis. In *Proceedings of the 26th annual international conference on machine learning.* 129–136.

[4] Ting Chen, Lala Li, and Yizhou Sun. 2020. Differentiable Product Quantization for End-to-End Embedding Compression. In *International Conference on Machine Learning.* 1617–1626.

[5] Ayushi Dalmia and Manish Gupta. 2018. Towards Interpretation of Node Embeddings. In *Companion Proceedings of the The Web Conference 2018.* 945–952.

[6] Yaroslav Ganin and Victor Lempitsky. 2015. Unsupervised Domain Adaptation by Backpropagation. In *International conference on machine learning.* 1180–1189.

[7] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. 2017. Improved Deep Embedded Clustering with Local Structure Preservation.. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence.* 1753–1759.

[8] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the Dimensionality of Data with Neural Networks. *science* 313, 5786 (2006), 504–507.

[9] Jyun-Yu Jiang, Patrick H Chen, Cho-Jui Hsieh, and Wei Wang. 2020. Clustering and Constructing User Coresets to Accelerate Large-scale Top-K Recommender Systems. In *Proceedings of The Web Conference.* 2177–2187.

[10] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A Contextual-Bandit Approach to Personalized News Article Recommendation. In *Proceedings of the 19th international conference on World wide web.* 661–670.

[11] Yijie Lin, Yuanbiao Gou, Zitao Liu, Boyun Li, Jiancheng Lv, and Xi Peng. 2021. COMPLETER: Incomplete Multi-view Clustering via Contrastive Prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 11174–11183.

[12] Jixiong Liu, Jiakun Shi, Wanling Cai, Bo Liu, Weike Pan, Qiang Yang, and Zhong Ming. 2017. Transfer Learning from APP Domain to News Domain for Dual Cold-Start Recommendation. In *Proceedings of the 1st Workshop on Intelligent Recommender Systems by Knowledge Transfer & Learning.* 38–41.

[13] Weiwei Liu, Xiaobo Shen, and Ivor Tsang. 2017. Sparse Embedded $k$-Means Clustering. (2017), 3319–3327.

[14] Hai Thanh Nguyen and Anders Kofod-Petersen. 2014. Using multi-armed bandit to solve cold-start problems in recommender systems at telco. In *Mining Intelligence and Knowledge Exploration.* Springer, 21–30.

[15] Keiichi Ochiai, Naoki Yamamoto, Takashi Hamatani, Yusuke Fukazawa, and Takayasu Yamaguchi. 2019. Exploiting Graph Convolutional Networks for Representation Learning of Mobile App Usage. In *2019 IEEE International Conference on Big Data (Big Data).* IEEE, 5379–5383.

[16] David Sculley. 2010. Web-scale K-means Clustering. In *Proceedings of the 19th International Conference on World Wide Web.* 1177–1178.

[17] Ying Shan, T Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. 2016. Deep Crossing: Web-scale Modeling Without Manually Crafted Combinatorial Features. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 255–262.

[18] Yuan Tian, Ke Zhou, Mounia Lalmas, and Dan Pelleg. 2020. Identifying Tasks from Mobile App Usage Patterns. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 2357–2366.

[19] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial Discriminative Domain Adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 7167–7176.

[20] Tongzhou Wang and Phillip Isola. 2020. Understanding Contrastive Representation Learning Through Alignment and Uniformity on the Hypersphere. In *International Conference on Machine Learning.* 9929–9939.

[21] Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. 2015. On Deep Multi-view Representation Learning. In *Proceedings of the 32nd International Conference on Machine Learning.* PMLR, 1083–1092.

[22] Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised Deep Embedding for Clustering Analysis. In *Proceedings of the 33nd International Conference on Machine Learning.* 478–487.

[23] Jie Xu, Huayi Tang, Yazhou Ren, Xiaofeng Zhu, and Lifang He. 2021. Contrastive Multi-Modal Clustering. *arXiv preprint arXiv:2106.11193* (2021).

[24] Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. 2017. Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering. In *International conference on machine learning.* 3861–3870.

[25] Junqi Zhang, Bing Bai, Ye Lin, Jian Liang, Kun Bai, and Fei Wang. 2020. General-Purpose User Embeddings based on Mobile App Usage. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 2831–2840.

[26] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. 2019. Recommending What Video to Watch Next: a Multitask Ranking System. In *Proceedings of the 13th ACM Conference on Recommender Systems.* 43–51.

## A DETAILED EXPERIMENTAL SETTINGS

This appendix provides detailed supplementary information for the preprocessing procedure of the proxy dataset, the formulation of the evaluation metrics and the hyper-parameter settings for baselines and DAUC.

### A.1 Preprocessing of the Proxy Dataset

To test the generalization ability of the proposed model, we built a proxy dataset with the Taobao Ad Display/Click Data. To simulate our task's knowledge transferring property from app usage to article reading behavior, we used the behavior on cheap and expensive items as the proxy data. Clicks on items cheaper than or equal to 160RMB were used as the proxy mobile app usage behavior, and clicks on items more expensive than 160RMB were used as the proxy article reading behavior. We applied latent semantic analysis (LSA) on the user-item display data and used items' latent vectors as the proxy article embeddings.

We also filter some cheap items and inactive users so that the proxy and Industrial datasets have similar task characteristics. On the one hand, long-tail cheap items are of little help for clustering but significantly increase the complexity of the model. Therefore, we filter cheap items with less than 50 clicks. On the other hand, users with very sparse click behavior in cheap items do not satisfy the settings of transferring knowledge from the information-rich domain to the target domain. So we filter users with less than 50 clicks on cheap items.

### A.2 Formulation of the Metrics

We show the formulation of the two metrics, i.e., Recall@$K$ and NDCG@$K$, as follows,

$$\text{Recall@}K = \frac{\sum_{i=1}^{K} rel_i}{|\mathcal{I}|},$$

$$\text{NDCG@}K = \frac{1}{R_Q}\sum_{i=1}^{Q}\frac{2^{rel_i - 1}}{\log_2(1+i)},$$

where $K$ is the size of the recommendation list, $rel_i = 0$ or 1 denotes whether the item at the rank $i$ is interacted in the test set or not, and $R_Q$ indicates the maximum possible cumulative component through ideal ranking. $|\mathcal{I}|$ is the number of all interacted items.

### A.3 Hyper-parameter Settings

There are many hyper-parameter settings for baselines and the proposed DAUC. To balance the efficiency and performance, we directly determine some of them based on our previous experience and find the optimal settings for the others according to the recommendation performance on the validation set in both datasets.

In terms of $k$-means, we use the mini-batch variant with a batch size of 10000 and set the maximum number of iterations as 100 for both datasets.

In terms of DEC and IDEC, the app behavior autoencoder is designed by fully connected layers. The behavior encoder is with the dimensions of $M$-800-600-400, and the behavior decoder is 400-600-800-$M$. The autoencoder was pretrained for 30 epochs with the reconstruction loss and then trained for 20 epochs with the clustering loss with a learning rate of 0.001. We set the coefficient of clustering loss as 0.1 for IDEC.

In terms of DCCAE, we use the same structures of the two autoencoders as our proposed DAUC. Namely, the two behavior autoencoders are designed by fully connected layers. The app behavior encoder is with the dimensions of $M$-800-600-400, and the article behavior encoder is with the dimensions of $N$-256-400 for both the datasets. The dimensions of the decoder and the corresponding encoder are in reverse order. The model is trained with a learning rate of 0.0001 for 30 epochs, and we saved the best model according to the recommendation performance on the validation set. We roughly experimented with different weights on the loss terms and found that equal weights for the reconstruction loss and the CCA loss worked well.

In terms of COMPLETER, we also use the same structures of the two autoencoders as our proposed DAUC. $\alpha$ is the regularization term of the entropy in the contrastive loss. We train several models when setting the $\alpha$ from 1 to 10. Then we generate and evaluate different versions of user clusters in the recommendation test on the validation set and determine the optimal $\alpha$ as 9. As for the dual prediction module, the encoders are fully connected layers with 400-256-400 dimensions for both datasets. We set both the weights of the reconstruction loss and the dual prediction loss as 0.1 after experiments with a different setting (0.01, 0.1, 0.2, and 0.5). The model is trained for 30 epoched, and the learning rate is 0.0001 for Adam. We added the dual prediction loss from the tenth epoch and saved the best model according to the recommendation performance on the validation set.

In terms of DAUC, $\tau$ is the temperature hyperparameter that influences the effect of the contrastive loss. We train several models when setting the temperature to 1.0, 0.5, 0.1, and 0.01. Then we generate and evaluate different versions of user clusters in the recommendation test on the validation set and determine the optimal temperature as 0.5. We set $\lambda = 0.5$ in the adversarial alignment for both datasets. The proposed DAUC is trained with a batch size of 128, and the learning rate is set to 0.0001 for Adam. We trained DAUC for 30 epochs and saved the best model according to the recommendation performance on the validation set.

## B MORE ANALYSIS OF THE OFFLINE EXPERIMENTAL RESULTS

We notice that cold-start users' offline metrics (i.e., Recall and NDCG) are higher than the overall scores on both datasets, which seems inconsistent with common intuitions as the cold-start recommendation is a more challenging problem. This is due to the limitation of the offline replay evaluation scheme. Cold-start users browsed fewer items in the past, and the algorithms know less about their preferences, so the recommender systems tend to recommend hot items in general. For example, in the Industrial dataset, the number of distinct exposed items by cold-start users is only about 65% of all distinct exposed items, and the number of distinct clicked items by cold-start users is only about 42% of all distinct clicked items. This makes it easier to guess offline what items a cold-start user clicked. Thus, the metrics seem to be higher. Also, the relative improvements on cold-start users and overall improvements should not be compared directly, due to the offline replay evaluations.