IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 19, 2020, accepted July 6, 2020, date of publication July 14, 2020, date of current version July 24, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3009253

Video Popularity Prediction: An Autoencoder Approach With Clustering

YU-TAI LIN[1], CHIA-CHENG YEN[2], AND JIA-SHUNG WANG[1], (Member, IEEE)
[1]Department of Computer Science, National Tsing Hua University, Hsinchu 30013, Taiwan
[2]Department of Computer Science, University of California at Davis, Davis, CA 95616, USA

Corresponding author: Jia-Shung Wang (jswang@cs.nthu.edu.tw)

This work was supported by the Ministry of Science and Technology, Taiwan, under Grant MOST 104-2221-E-007-017.

ABSTRACT Autoencoders implemented by artificial neural networks (ANNs) are utilized to learn the latent space representation of data in an unsupervised manner, and they have been widely used in recommender systems. For instance, several collaborative denoising autoencoder (CDAE) models have shown that their performance gains outperform that of the collaborative filtering based (CF-based) models. In this work, a near-optimal Top-K forecasting solution is proposed for our advanced autoencoder recommender systems. We propose a method which utilizes CDAE model in predicting the Top-K popular videos in an upcoming time period. In order to improve the prediction accuracy, we also propose an autoencoder based recommendation algorithm with the help of K-means clustering that upgrades the performance of the original autoencoder model. The experimental results show that our method increases significantly the Average Precision (AP) and Recall values by nearly 30%. We then further utilize our proposed autoencoder model with clustering in predicting Top-K popular videos. The applications of predicting Top-K popular videos can be used in the video delivery for the Mobile Edge Computing (MEC) environment to avoid bottleneck in the constricted capacity of backhaul link. Namely, the performance gain will be upgraded if our proposed method precisely predicts and caches the Top-K popular videos in advance with the help of a better forecasting model.

INDEX TERMS Top-K ranking and predicting, autoencoder, caching, K-means.

I. INTRODUCTION

To cope with rapid development on e-commerce and online video platforms, recommender systems become a very important way of locating target users and producing e-commerce recommendations. According to historical transactions of users and items, methods of recommender systems can suggest a specific set of items for each individual user in a personalized way. This can be beneficial to diversified users and items waiting for recommendations. In general, methods for recommender systems can be roughly categorized into three ways [1], i.e. content-based methods, collaborative filtering based (CF-based) methods and hybrid-based methods that merge content-based and CF-based methods.

The content-based method takes a user's profile or item's description into account in order to recommend target items with similar properties. On the other hand, the CF-based method utilizes a user's previous behavior, i.e. user's rating or purchase history and preference data from other users, to make recommendations. In addition, the hybrid-based method combines content-based and CF-based methods, and benefits from their advantages when making recommendations. In [2], they combined traditional CF i.e. probabilistic matrix factorization (PMF) [3] with topic modeling. In [4], they used recurrent neural network (RNN) to capture items' descriptions and then use CF to make predictions.

Currently, the wireless communication has become more and more important because people increasingly rely on cell phones for getting information and recreations. As mentioned in [7], global mobile data traffic will continue growing at a compound annual growth rate (CAGR) of 54 percent from 2016 to 2021. Mobile video traffic accounting for over three-fourths (78 percent) of total mobile data traffic is expected by 2021. To deal with such large amount of mobile data traffic, heterogeneous and small cell networks (HetNets) [8] has become a solid solution. However, even with the help of HetNets, we still face a bottleneck which is a limited capacity of backhaul link to the core network.

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

Thus, the emergence of Mobile Edge Computing (MEC) [9] technology offered a feasible way in which data can be pre-processed and cached on the edge side (such as small cells). With the help of MEC, the buffered videos can be delivered by caches with less network latency. Moreover, network traffic load can be reduced by caching the most popular video clips on the edge side. Consequently, predicting the Top-K popular videos in an upcoming time period and pre-caching them on the edge side has become an important subject to be solved.

In this work, we first attempt to predict the Top-K popular videos in an upcoming time period with an innovative recommendation algorithm - the collaborative denoising autoencoder (CDAE) [5], [6]. In our experiments, the results show that the CDAE model outperforms the baseline model – expert-based model [20]. However, we are not satisfied by the results. Thus, we focus on improving the accuracy of CDAE with the help of clustering. We found that by clustering users with similar preferences into same groups and fine-tune the CDAE model for each group, we can improve the CDAE model to better fit for each group. In the first step, we train the original autoencoder model to reveal user-specific vectors (user-specific vector or feature vector). Then, similar users can be grouped into the same cluster by means of K-means with their user-specific vectors. Afterward, for each cluster, fine-tune the trained autoencoder model separately. Finally, utilize each fine-tuned model to make recommendations. The experimental results show that our method increases significantly the Average Precision (AP) and Recall values by nearly 30%. We then back to subject of predicting the Top-K popular videos by applying our new improved model. The results exhibit a significant improvement in all experiments.

The contributions of this paper are summarized as follows:

1) Propose a refined autoencoder model using a clustering method to improve the prediction accuracy.
2) Adopt our proposed model in predicting potential (recent) Top-K popular videos and propose an ensemble method to calculate the prediction of preference of potential Top-K popular videos.
3) Propose a weighted cross entropy loss function to boost prediction performance in potential Top-K popular videos

The rest of this paper is organized as follows. We briefly elaborate CF-based recommender systems, deep learning in recommender systems, and cluster-centric small cell networks in Section II. Our proposed method which combines autoencoder with clustering and the prediction of the Top-K popular videos in an upcoming time period is presented in Section III. The experimental results are revealed and discussed in Section IV. Finally, conclusion and future works are drawn in Section V.

II. RELATED WORK

A. COLLABORATIVE FILTERING BASED RECOMMENDER SYSTEMS

Collaborative Filtering based (CF-based) methods can be divided into two categories: memory-based CF and model-based CF. Memory-based CF, also known as neighborhood-based CF, makes recommendations via similarity which explores the relationship among users and items. At first, it analyzes user-item matrix to identify the relationship among items (or users), and calculates the similarity through associations, then makes a prediction. In [10], they analyzed various memory-based CF algorithms especially for item-based CF, and compared them to the basic K-nearest neighbor approach. According to the experimental results, they suggested that item-based CF algorithms provide dramatically better performance than user-based CF.

On the other hand, model-based CF makes recommendations by modeling ratings. Such models are built using various data mining and machine learning algorithms to make recommendations. In [3], they used low-dimensional model to model a rating matrix, i.e. assuming that a rating matrix R can be factorized into multiplication of two low-dimensional matrices, $R = U^T V$. Then, utilize gradient descent to update model to learn factorization feature. In [5], they proposed a CF model based on the autoencoder paradigm. First, autoencoder is used to encode user rating vector into low-dimensional vector and then, this low-dimensional vector will be decoded back to rating vector. The key idea of autoencoder applied to recommender systems is that it can learn important rating features from users through autoencoder. In [6], they also applied autoencoder to recommender systems. Compared to [5], in their model, a learnable *user-specific vector* for each user is included to improve the quality of recommendation.

In addition to historical interactions of users and items, recently, the side information has also been considered to avoid the sparse matrix problem and aid recommender systems. The knowledge graph (KG) which consists of nodes (items) and edges (relations) was utilized as the side information in [11]. KG is able to diversify recommended candidates by consisting diverse types of relations and increase understanding to recommender systems by connecting a user's records and candidate items. Their proposed RippleNet which discovers high-order preferences of users on KG took a user-item pair, and then generated the probability that this user would like to select this item. The RippleNet was applied to several types of recommendations and revealed a better performance compared to cutting-edge baselines.

B. DEEP LEARNING MEET RECOMMENDER SYSTEMS

With thriving of deep learning development in recent years, deep learning related applications and experiments have been conducted in many research fields. Similarly, deep learning techniques can be suitably applied to recommender systems. In [4], RNN was used to capture descriptions of users or items and combine them with rating vectors to produce recommendations. When it comes to CF-based methods, there are plenty of deep learning implementations, such as autoencoder and RNN. In [12], they used autoencoder to encode user rating into hidden layer and employed CF to calculate user similarity via input rating and hidden layer features.

Then, predict rankings through decoding output of autoencoder and CF similarity. In [13], RNN was applied to model a user's web session history, and then make recommendations from the session recordings. Also, recommender networks using RNN to capture temporal dynamics was proposed in [14]. That is to say, they fed ratings of each time period into RNN for training and employed RNN to predict ratings in an upcoming time period. Their experimental results showed that the RMSE loss is better than state-of-the-art methods with the help of modeling the rating temporal dynamics.

Recently, a deep item-based CF [15] utilized deep neural networks to further consider the nonlinear and higher-order relationships such as the same genre, actor, director among items. This approach extracted a similarity (second-order relation) between two items' embeddings. Then, they employed nonlinear neural networks to learn from these second-order relations and detect the interactive information (higher-order relation) between any two items which have interacted. Their results showed that this approach can model relations among movies more effectively.

Moreover, in [16], they targeted on two demerits of multi-mode factor models: failure of capturing nonlinear relations between items, and difficulty of ensuring diversity of recommendations. They proposed a tensor-based approach using the Bayesian algorithm and deep neural network for video recommendations. Their proposed Deep Canonical PARAFAC Factorization (DCPF) clustered the multi-mode data along different dimensions. Then, in order to extract a more robust representation of latent relationships of each mode for the multi-mode data, a multi-layer factorization was applied to the matrix. The results demonstrated that the proposed approach can capture the latent patterns well.

C. CLUSTER-CENTRIC SMALL CELL NETWORKS

Cluster-centric small cell networks (SCNs) [17] grouped small cells into disjoint clusters to utilize cache space of every small cell. However, the capacity of the cache space for each SCs is limited. The backhaul transmission of video data would become a bottleneck due to frequently cache replacement. Thus, in order to mitigate this problem, predicting the most popular videos and caching them in advanced are crucial. In our previous work [18], a clustering method was used to separate users with a similar preference into the same group, and then assign groups to set of small cells. We also shared the caching space among cooperated SCs with the help of distributed LT codes. The simulation showed that the backhaul traffic rate can decrease from 38% to 10% if cache space is acceptable, and decrease from 64% to 33% if cache space is poor.

III. THE PROPOSED METHODS

With the user ratings and help of the CDAE [6], the proposed methods predict potential Top-K videos in an upcoming time period. The objective of autoencoder is to train the model that minimizes the difference between the input and the predicted output. In the training stage, the autoencoder model learns the latent factor of each user by transforming his/her input rating into a lower dimensional feature vector, and then predicts what this user might also prefer. However, preferences of users have not been considered by previous studies yet. We assume the preference of each user plays a crucial role in predicting potential Top-K videos in an upcoming time period. Hence, we take users' preferences into account by adding the latent factor of each user with a preference vector V_u, updated by the optimizer, which minimizes the weighted loss as shown in Fig. 1. In addition, in our refinement stage, users are classified by their preference vectors V_u to further improve the prediction of potential Top-K videos. Note that all symbols are summarized in Table 1.

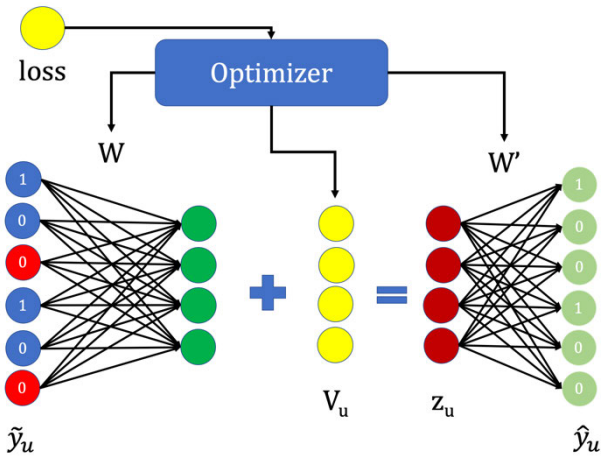

FIGURE 1. Collaborative denoising autoencoder model (CDAE) [6].

A. POTENTIAL TOP-K POPULAR VIDEOS PREDICTION

According to the video viewing experience, the most popular videos either on YouTube or Netflix are fairly static for a while. Most of them have been on the popular list for weeks. This indicates that most people are likely to have the same taste of videos for a short period of time. Based on this observation, we define a certain time period and use CDAE [6] to acquire users' preference at that time period. By aggregating the prediction of users' preferences, we are able to infer Top-K popular videos in an upcoming time period. As demonstrated in the experimental results, this Top-K forecasting method performs near-optimal for four datasets in the sense of *still-in* which will be discussed in sub-section IV.B. In the following sub-section, we elaborate how to predict Top-K popular videos by our weighted loss function and CDAE [6].

1) COLLABORATIVE DENOISING AUTOENCODER (CDAE)

As in [6], they proposed an autoencoder model which is consisted of 2 fully connected layers and learnable user-specific vectors V_u. V_u represents preference distribution of user u, see Fig. 1.

Let I be a user set and J be an item set, and $Y^{I\times J}$ be a history matrix for all user-item ratings. Based on $Y^{I\times J}$,

TABLE 1. Summary of notation.

Symbol	Definition
I	A user set
J	An item set
$Y^{I\times J}$	A user-item rating history matrix
V_u	User-specific vectors which represent preference distribution of each user
z_u	A hidden layer transformation function
$\tilde{y}_u$	The input rating vector of items by user u
$\hat{y}_u$	The prediction vector of items for user u
$\tilde{y}_{ui}$	The input rating of item i by user u
$\hat{y}_{ui}$	The prediction of the preference of user u on item i
u_i	User i
v_j	Item j
N_u	The average number of users who have viewed videos across time periods
N_v	The average number of items which have been viewed across time periods
C_v	The average number of average view counts for each item by different users across time periods
Y'	A $N_u \times N_v$ likelihood score matrix
k_j	The upper quartile score of video j
S_j	A set of the scores of the video j which are predicted by different users and greater than k_j
a_j	Average score of S_j
l_i	Training loss of user i
y_j	True rating of user i to video j
$\hat{y}_j$	The prediction of user i on video j
w_j	The weight of video j
M_c	The number of clusters
V_c	A viewing count set of the cluster's viewed items
r_{cj}	The rank of item j in cluster c
n_c	The number of users in cluster c
n	The number of total users in training time periods

each user's rating $\tilde{y}_u = \{y_{u1}, y_{u2}, y_{u3}, \ldots, y_{uJ}\}$ can be used as an input to the model. In [6], they used a fully connected layer to transform user-item ratings into the hidden layer. Then, for each user, user-specific vector V_u was added to the corresponding hidden layer vector individually. The input to hidden layer transformation function is listed as below:

$$z_u = h(W^T \tilde{y}_u + V_u + b) \tag{1}$$

where $h(\cdot)$ is an activation function, W^T is transformation matrix, and b is a bias. Finally, the second fully connected layer was used to transform latent vector into output vector. The transformation function from hidden layer to output layer is listed as below:

$$\hat{y}_u = f(W' z_u + b'_u) \tag{2}$$

where $f(\cdot)$ is also an activation function and W' is transformation matrix from hidden layer to output layer and b'_i is a bias term.

By optimizing transformation matrix W, W', and user-specific vector, the model can learn preference of each user from their viewing history and forecast what user would prefer in the future.

2) PREDICTION OF TOP-K POPULAR VIDEOS BY CDAE

For each dataset, we have users' viewing history including user id, item id (which items he/she has viewed) and the timestamp (when he/she viewed). The format of each dataset is expressed as a list of tuples given by

$$< u_i, v_j, timestamp >$$

where u_i denotes user i, and v_j denotes item j. Then, we separate rating history into training set and prediction set consecutively. The goal is to use the rating history before *separation point* to predict Top-K videos in an upcoming time period. As shown in Fig. 2, we separate time series data into 3 parts – training part, prediction part and rest part by a separation point (the dash line in Fig. 2).

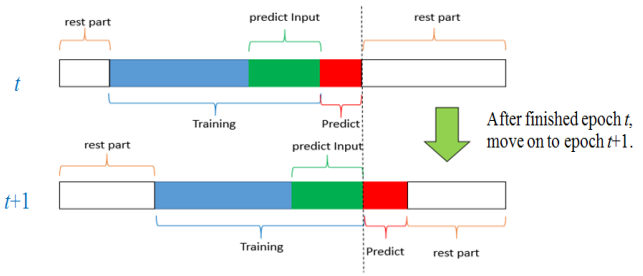

FIGURE 2. The separation point, training and testing (prediction) time periods.

In the training stage, a fixed interval of viewing history (blue and green parts in Fig. 2) is selected for training an autoencoder model. The latest three time periods in the training set are selected as prediction inputs (the green part in Fig. 2). After training, the trained model is utilized to predict the potential Top-K popular videos in an upcoming time period (red part in Fig. 2). Since the observation mentioned above, the users' preference remains unchanged for a while. Algorithm 2 selects Top-K videos which are predicted by all users, and summarize the Top-K popular ones as the most popular videos in an upcoming time period. This procedure will be executed continually for the coming time periods. The detail of procedure is shown in Algorithm 1.

Note that instead of including all rating data before any separation point into the training set, we only collect data from a few time periods as the training set. The reason is that, although the Top-K popular videos within each time period remain fairly constant, they are still quite dissimilar from the long-term perspective. If we involved all rating data into the training set, the autoencoder model would be seriously disturbed as time moving forwards. Also, we do not pick initial of separation point too early since it could lead to a lack of training data and jeopardize the overall performance of the model prediction.

In practice, a typical user usually rates a relatively small fraction of movies. In both Table 2 and Fig. 3, the sparsity, i.e. (view count) / (# of users $*$ # of items), is prevalent in every time period. The sparsity means that even for the Top-K popular videos, videos would be viewed by a small

Algorithm 1 Next Time Period Prediction

Input: List of tuples $< u_i, v_j,$ timestamp$>$ of the rating history.

Output: Top-K popular videos for each time period.

1. Initialize separation point set S.
2. **For** each separation point in S do:
3. Separate data into training set and testing set.
4. Train the autoencoder model with training set.
5. Take the latest 3 time periods rating history in training set as the input for prediction.
6. Use trained autoencoder model to predict potential Top-K popular videos for each user in an upcoming time period.
7. Select Top-K popular videos from all users (Algorithm 2).
8. Move on to the next time period.
9. **End For**

TABLE 2. Average sparsity across time period.

	N_u	N_v	C_v	Avg. sparsity
ITRI	220	885	5	2.07%
Netflix(small)	767	74	17	3.36%
Netflix(1 Year)	60,699	8,330	148	0.24%

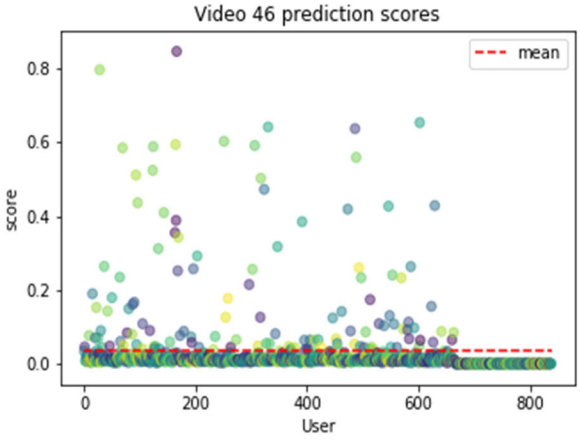

FIGURE 3. Prediction score of video No. 46, Netflix(small) dataset in a time period.

number of users in each time period. Note that Fig. 3 shows prediction scores of video No. 46 in a time period and not every user involved in that time period; thus, the prediction result shows that some users assign 0 as their rating scores to video No. 46. Table 2 lists average sparsity of three different datasets across time periods. N_u denotes the average number of users who have viewed videos across time periods, N_v denotes the average number of items which have been viewed across time periods, and C_v indicates the average number of average view counts for each item by different users across time periods. Also, the prediction score of Video #46 using an autoencoder model in a time period is depicted in Fig. 3. The mean (average score) is quite low because only a few users gave the ratings for Video #46; thus, quite a few prediction scores above some threshold, say 0.4.

Due to the sparsity issue mentioned above, our method is devised to choose the Top-K popular videos in a time period. Algorithm 2 shows the detail of selecting the Top-K popular videos. Based on the autoencoder model with the training time periods (see Fig. 2), we compute a $N_u \times N_v$ likelihood score matrix Y' of the current time period (red period in Fig. 2). Let Y' be a score matrix where all elements are between 0 and 1: $\{0 \leq \hat{y}_{ij} \leq 1 | \forall\, 0 \leq i < N_u, 0 \leq j < N_v\}$. The predicted likelihood, $\hat{y}_{ij}$, indicates the probability that video j would be watched by user i, i.e. if user i is predicted more likely to watch video j, the prediction score is much closer to 1 and vice versa. Let k_j be the upper quartile score of the video j. Let S_j be a set of the scores of the video j which are predicted by different users and greater than k_j. Then, compute a_j for video j by averaging all scores in S_j. Finally, select the K videos with the highest a_j as the Top-K popular videos for the next time period. Fig. 4 shows the recalls for various quartile values (e.g., 50%, top 25%, and top 15%) by using three different models and datasets, which will be discussed in the next section.

Algorithm 2 The Selection of Top-K Popular Videos

Input: The (predicted) likelihood score matrix ${Y'}_{N_u \times N_v}$.

Output: Indices of Top-K popular videos.

1. **For** each item in the current time period:
2. Assign k_j to be the upper quartile score of item j.
3. Generate Set, $S_j = \{s_{j1}, s_{j2}, \ldots, s_{jM} | \forall s_{ji} s_{ji} > k_j\}$.
4. Compute the average a_j of S_j, $a_j = \sum_{n=1}^{M} S_{jn}/M$.
5. **End For**
6. Select these videos that have the highest K scores of all a_j to be Top-K popular videos.

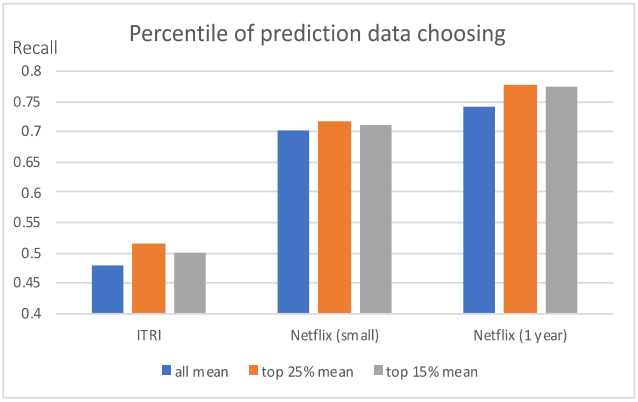

FIGURE 4. The recalls for various quartile values (e.g., 50%, top 25%, and top 15%) by using three different models and datasets.

In our model applied to Top-K popular videos prediction, we choose cross entropy loss [6] as our loss function. Moreover, in order to boost the prediction performance of

Top-K popular videos, we propose a weighted loss function l for optimization and is given by

$$l\left(\tilde{y},\hat{y}\right)_i=\sum_{j=1}^{J}-\tilde{y}_j\times log\left(\hat{y}_j\right)-\left(1-\tilde{y}_j\right)\times log(1-\hat{y}_j)\times w_j \tag{3}$$

where l_i is the training loss for the user i, $\tilde{y}_j$ is the actual rating given by the user i to the video j, $\hat{y}_j$ is the predicted rating given by the user i to the video j, and w_j is the weight value for the video j. As exploiting weighted cross entropy loss function, the proposed model is capable of focusing on videos which are expected at higher rank in the training phase.

B. PREDICTION REFINEMENT

With CDAE [6], the prediction performance shown in sub-section IV.D.3 can be refined by clustering users. In the following sub-section, we introduce the key concept of CDAE – the user-specific vector. We elaborate how it represents individual preference for each user in III.B.1, and how our proposed clustering refinement method improves the prediction accuracy. We introduce how to apply the proposed refinement method to predict the top-K potential popular videos in next time period in III.B.2.

1) USER-SPECIFIC VECTOR AND CLUSTERING REFINEMENT

As mentioned above, the difference between CDAE [6] model and original autoencoder model [5] is the additional user-specific vector which represents preference distribution of each user. The benefit of adding user-specific vector was shown in the experiment in [6]. However, we find that the performance can be improved by clustering techniques. Since the original CDAE model (non-clustering) was trained to learn all users' preferences, it is unable to well fit for a particular user with a specific preference. Thus, we cluster users with a similar preference into the same group and then, fine-tune the trained CDAE model with the users' viewing histories in the same group. The fine-tuned model will be better fit for the users in the group.

In initialization stage, we use Gaussian distribution to initialize the user-specific vector V_u by a M-dimensional vector, which has the same dimension as the hidden layer. We, then, apply them to the autoencoder model according to each user's identification. In the update stage, the user-specific vector is fine-tuned by the optimizer to enable the vectors to have a better representation of user's feature of rating distribution.

Fig. 5 shows the similarity of each user-specific vector. X-axis lists 4 pairs of users (by user ID) from which their ratings are totally different. That is to say, we believe that the pairs of users have different tastes of videos. Y-axis represents cosine similarities. As shown in Fig. 5, their user-specific vector similarities drop after training. It shows that the CDAE model can be truly trained and updated the user-specific vectors to better fit users' characteristics.

As mentioned above, a user-specific vector represents preference distribution of each user. Hence, we improve the performance in prediction accuracy by 1) clustering users by their preference, and 2) refining the model by each group. In our work, we utilize the K-means algorithm to categorize trained user-specific vectors (user-specific vector or feature vector) and group similar users to boost the performance of the original autoencoder model [5]. K-means clustering [19] is one of the most popular unsupervised learning algorithms so far. K-means can categorize similar users or items into same clusters via user/item similarity. The K-means algorithm divides a set of n samples into several clusters. This clustering algorithm aims to choose centroids μ that minimizes within-cluster sum of squared criterion:

$$\sum_{i=0}^{n}\underset{\mu_j\in c}{min}(\left|\left|x_i-\mu_j\right|\right|^2) \tag{4}$$

The process is listed in Algorithm 3 and the corresponding flowchart is shown in Fig. 6. Note that we adopt different K values to K-means based on the size of users in different datasets. The detail of how to determine K values will be elaborated in section IV.C. First, we pre-train CDAE model by using user-item rating history as an input in order to get the trained d-dimensional user-specific vector V_u. Then, apply all user-specific vectors V_u into K-means procedure to organize similar users into groups. Finally, for each group, we fine-tune the pre-trained autoencoder model to well fit each group's rating distribution. In our experimental results

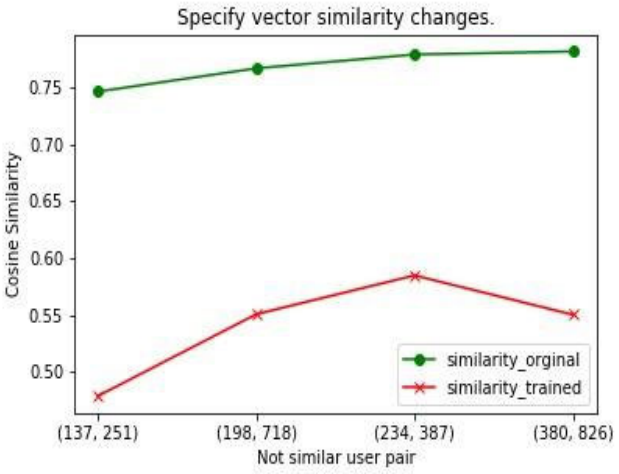

FIGURE 5. A demonstrated example of changing of similarity with user-specific vectors (Netflix small dataset).

Algorithm 3 Clustering-Based CDAE

Input: User-Item rating history matrix $Y^{I\times J}$
Output: Fine-tuned autoencoder models for all groups
1. Pre-train CDAE by using $Y^{I\times J}$ to train user-specific vectors V_u
2. Organize similar users into groups using user-specific vectors V_u by $K-$ means.
3. **For** each cluster:
4. Fine-tune pre-trained model to fit each group's rating distribution.
5. **End For**

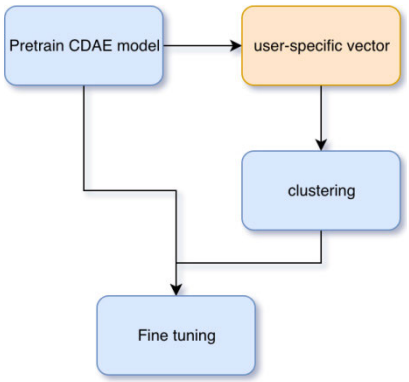

FIGURE 6. The flowchart of the proposed autoencoder model with clustering.

discussed later, it shows that the performance of our proposed method is promising for various scale of datasets.

2) PREDICTION OF CLUSTERING BASED TOP-K POPULAR VIDEOS

In this section, we introduce prediction of Top-K popular videos by CDAE with clustering refinement. For potential Top-K videos, our clustering-based prediction approach consists of two procedures (Algorithm 2 and Algorithm 3). For each separation point, the autoencoder model is trained with appropriate time periods (see Fig. 4). Then, apply the K-means algorithm to categorize users by their trained user-specific vectors. After clustering, we fine-tune the trained autoencoder model and utilize each cluster's autoencoder model to select their potential Top-K popular videos. The prediction results are several lists of Top-K popular videos for all clusters. Eventually, all of these lists are combined by an ensemble method in order to formulate the final Top-K popular videos.

In general, view counts of videos and their popularity ranking should follow the Zipf's distribution if the scale is large enough. Fig. 7 shows our dataset's popularity distribution (log-scale). Unfortunately, we observe that the popularity distribution at every time period does not strictly follow Zipf's distribution because the number of viewed videos (delivered rankings) at every time period is very insufficient. Only in Netflix (1 Year) dataset, the number of watched videos at every time period is adequate to enable the popularity distribution to follow Zipf's distribution. Our ensemble method integrates all of lists by using Eq. (5) and Eq. (6) to calculate score of Top-K popular videos from each cluster. That is to say, if the number of viewed videos at a time period is adequate (larger than K), then we use the Zipf's distribution to calculate the unified scores. Otherwise, we have to use the popularity distribution of the training set of the cluster to calculate scores if the number of viewed videos at a time period is insufficient.

$$score_j = \sum_{c=1}^{M_c} \alpha_c \times z(r_{cj}) \times \frac{n_c}{n} \tag{5}$$

$$z(r) = \begin{cases} \frac{1/r^{\rho}}{\sum_{k=1}^{N'} 1/k^{\rho}}, & if\,|V_c| > K \\ \frac{v_{cr}}{\sum_{k=1}^{|v_c|} v_{ck}}, & if\,|V_c| \leq K \end{cases} \tag{6}$$

M_c denotes number of clusters, r_{cj} denotes rank of item j in cluster c, n_c denotes the number of users in cluster c, n denotes the number of total users in training time periods, and $z(\cdot)$ denotes ensemble score function for item j in rank r. $V_c = \{v_{c1}, v_{c2}, \ldots, v_{ck} | v_{c1} \geq v_{c2} \geq \ldots \geq v_{ck}\}$, is a viewing count set of the cluster's viewed items. In such a way, we can come up with an overall Top-K popular videos from every cluster in this time period.

Another challenge for predicting the Top-K popular videos in an upcoming time period is the unknown (first appearance) videos. It is difficult to predict the new coming videos since we definitely have no future knowledge. Thus, in case of some videos being so popular that will be in Top-K at first appearance, we could not forecast them eventually. Fig. 8 shows the ratios of new coming videos in Top-K lists for three datasets. We can observe that there is rather small ratio of new coming videos in Netflix (small) and Netflix (1 Year) compared to ITRI dataset. Since videos in Netflix are all movies or TV series and both of them cannot be produced

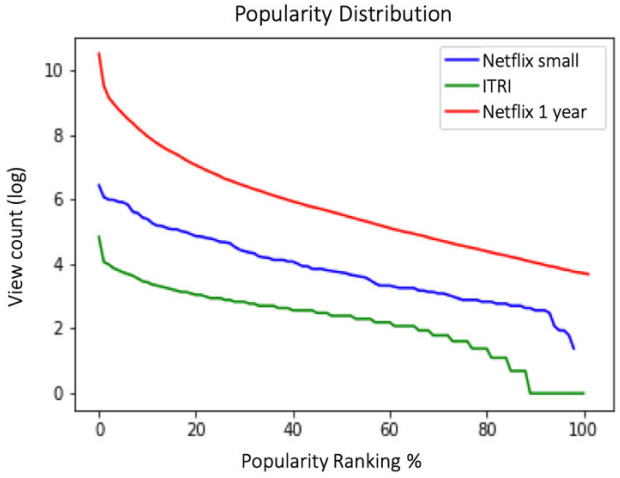

FIGURE 7. Popularity distribution of 3 datasets.

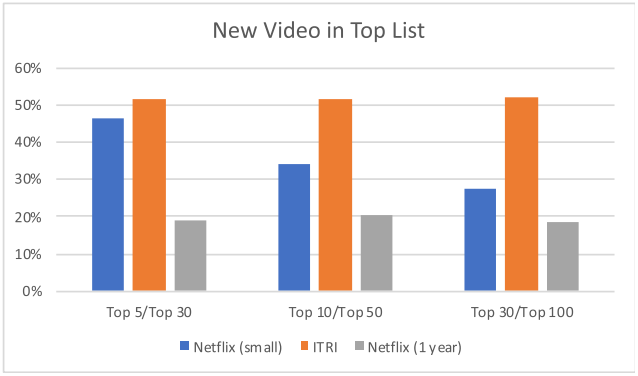

FIGURE 8. The ratios of new videos in Top-K lists for three datasets. K = 5, 10, and 30 for Netflix (small) and ITRI. K = 30, 50, and 100 for Netflix (1 year).

in a short-time, the list of top videos remains unchanged for a while. On the other hand, the ratio of new coming videos in ITRI dataset exceeds 50% because ITRI data are traces of some pilot runs with a relatively small number of users. There are many new videos coming over time even though some of them will quickly disappear from the ranking. Hence, the ranking of videos changes at relatively rapid rates.

IV. EXPERIMENTAL RESULTS

In order to highlight the improvement of our proposed method, our experimental evaluation is twofold: (1) the improvement of the clustering method compared to the original collaborative autoencoder [5] and (2) the accuracy of prediction for the potential Top-K popular videos. In this section, the datasets and evaluation metrics are introduced in sub-sections IV.A and IV.B. Next, we compare results of the clustering method to the original model in sub-section IV.C. Finally, the experimental results of potential Top-K popular videos are shown in sub-section IV.D.

A. DATASETS

We use four datasets in our experiments: ITRI, Netflix (small), Netflix (1 Year) and Netflix (large). ITRI has real traces of some experimental runs, which were collected by the Industrial Technology Research Institute (ITRI), Taiwan. All Netflix datasets are from Netflix Prize datasets. We created a small scale Netflix (small) dataset from Netflix data with a small group of users, and selected a whole year (2005) to create Netflix (1 Year) data for evaluating the proposed Top-K popular videos prediction. We converted Netflix dataset into $< 0, 1 >$ representation where 1 denotes that a user has seen this video and 0 indicates that a user has not seen it yet. This transformation is widely used in previous works (recommender systems) [8], [19] as an implicit feedback. The detail of datasets is listed in Table 3.

TABLE 3. The four test datasets.

	# of users	# of items	# of ratings	sparsity	time elapsed
ITRI	276	989	14072	5.16%	06/19/2016 ~ 11/14/2016
Netflix (small)	839	99	8709	10.49%	01/16/2000 ~ 12/31/2005
Netflix (1 Y)	96,684	9,394	9,460,526	1.17%	01/01/2005 ~ 12/31/2005
Netflix (large)	256,683	13,590	42,656,842	1.22%	02/28/2004 ~ 12/31/2005

B. EVALUATION METRICS

Precision, Recall as well as Mean Average Precision (MAP) are widely recognized as the classic evaluation metrics of recommender systems. Precision and Recall suffer from not being able to evaluate the ranking (prediction) of recommended items. The items that have been adopted should be recommended in a higher rank and those which have been unfavorable should be in lower ranks. Accordingly, MAP is recognized to not only evaluate how many items are truly adopted by users in the recommending list but also calculate their rankings.

To assess the refined autoencoder with clustering, we follow the prediction result choosing procedure in [6] where the fine-tuned autoencoder model would recommend each user with Top-N highest predicted value items, not including the training set. The performance is evaluated by using Recall@N and AP@N where $N = 5$ and 10, which means the N most videos we predict the user would watch for each user.

On the other hand, to assess the performance of our proposed model which predicts the popular videos in an upcoming time period, the prediction accuracy of Top-K popular videos in an upcoming time period is evaluated by using Recall@K and Still_in_Recall@K where $K = 5$, 10, 30, 50, and 100.

1) PRECISION AND RECALL

Given a Top-N recommendation list $C_{N,\,rec}$ predicted by each user, precision and recall is denoted by

$$Precision@N = \frac{|C_{N,rec} \cap C_{true}|}{N} \tag{7}$$

$$Recall@N = \frac{|C_{N,rec} \cap C_{true}|}{|C_{true}|} \tag{8}$$

where $C_{N,\,rec}$ is the Top-N videos with the highest predicted scores of each user and C_{true} represents the set of videos that user adopted in testing data.

2) MEAN AVERAGE PRECISION (MAP)

Average Precision (AP), as mentioned, can evaluate not only hit ratio of recommendation but also the video's ranking of recommendation. AP@N gives a better credit if videos are recommended in higher rank appropriately and is defined by

$$AP@N = \frac{\sum_{m=1}^{N} Precision@m \times rel(m)}{min\{N, |C_{true}|\}} \tag{9}$$

where $Precision@m$ are the precision of Top-M highest ranked item in the recommendation set $C_{N,\,rec}$, $rel(m)$ equals to 1 if the item at rank m is adopted; otherwise, $rel(m)$ is 0, and MAP@N is denoted as mean of AP scores for all users.

3) STILL-IN RECALL

For the prediction of Top-K popular videos in an upcoming time period, it is difficult to predict the new coming (first appearance) videos since we definitely have no priori knowledge to forecast them. Thus, in case of some videos being so popular in Top-K at first appearance, we propose a probable fair index, called Still_in_Recall@K to gauge the reasonable recall score. Still-in means that a video is currently in the Top-K popular list and will remain static in next time period. The Still_in_Recall@K is defined by

$$Recall_S@K = \frac{|C_{K,rec} \cap (C_{present} \cap C_{next})|}{|(C_{present} \cap C_{next})|} \tag{10}$$

where $C_{K,\,rec}$ represents the prediction of Top-K popular videos in an upcoming time period, $C_{present}$ denotes real Top-K popular videos in the current time period, and C_{next} denotes real Top-K popular videos in an upcoming time period. Assume that current Top-5 videos are $\{a, b, e, g, i\}$, Top-5 videos in an upcoming time period are $\{a, b, e, d, g\}$, and the predicted Top-5 videos are $\{a, e, g, f, k\}$. Then, the still-in recall is calculated by

$$Recall_S@5 = \frac{|\{a, e, g, f, k\} \cap \{a, b, e, g\}|}{|\{a, b, e, g\}|} = 0.75$$

4) F1-SCORE

For the prediction of Top-K popular videos in an upcoming time period, we also apply F1-Score to evaluate the model's performance. As mentioned, recall and precision focus more on positive and negative samples, respectively. In contrast, F1-score combines both of them to measure the overall performance. F1-score@K is defined by

$$F1@K = \frac{2 \times Recall@K \times Precision@K}{Recall@K + Precision@K} \tag{11}$$

C. CLUSTERING BASED CDAE

1) DATA DESCRIPTION AND INITIAL SETTING

Three datasets: ITRI, Netflix (small) and Netflix (large) datasets are considered for examining refined models in this section. In ITRI and Netflix (small) datasets, the users are categorized into 10 clusters because the number of users is relatively small. In Netflix (large) dataset, users are classified into 50 clusters. The activation functions in Eq. (1) and (2) are sigmoid function. The cross entropy is used as our loss function. In clustering procedure, the user-specific vector and feature vector are applied to organize the set of users for comparison.

2) REFINEMENT RESULT IN AP AND RECALL

Fig. 9 to 16 show performance comparisons between non-clustering CDAE model [6] and two different clustering refinements (user-specific vector and feature vector) in terms of AP@N and Recall@N. In this section, the comparisons are threefold: (1) small and large datasets, (2) clustering and non-clustering, and (3) user-specific vector and feature vector.

First, ITRI and Netflix (small) can achieve higher improvement because the distribution of users within the same cluster in smaller datasets is close to uniform. In ITRI and Netflix (small), the users within the same cluster are more uniform than the users within the same cluster in Netflix (large); therefore, the performance of fine tuning of both datasets are consequently better than that of Netflix (large). Second, feature vectors can extract more information from user's rating distribution than user-specific vectors because they are capable of combining users' rating (input) and the corresponding user-specific vectors. Hence, the clustering refinement with feature vector has a better performance than that with user-specific vector.

In Fig. 10, it can be easily observed that in small datasets, i.e. Netflix (small) and ITRI, the AP@5 of two clustering refinements can be improved by around 20%. Even in Netflix (large) dataset, there is about 5% improvement by using feature vector. Fig. 11 to 16 show AP@10, Recall@5, Recall@10 of three datasets and their improvements by using different clustering refinements. Apparently, our proposed model performs better than the original model [5]. As shown in Fig. 12, in the ITRI dataset, the improvement can be increased by 25%, and in the Netflix (large) dataset, it can be

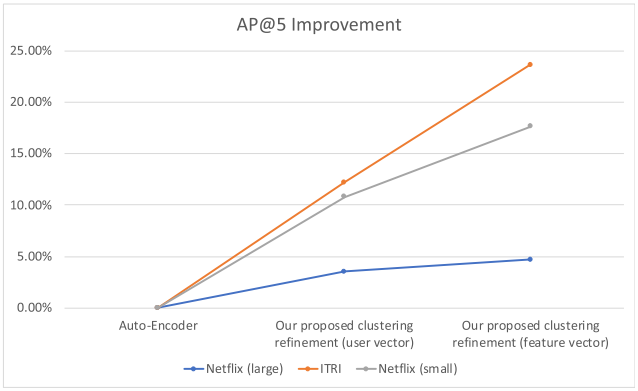

FIGURE 10. Improvements of AP@5 between three models.

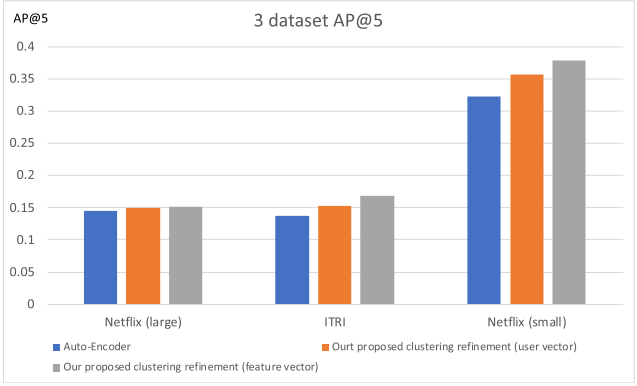

FIGURE 9. The comparison of AP@5 between three models.

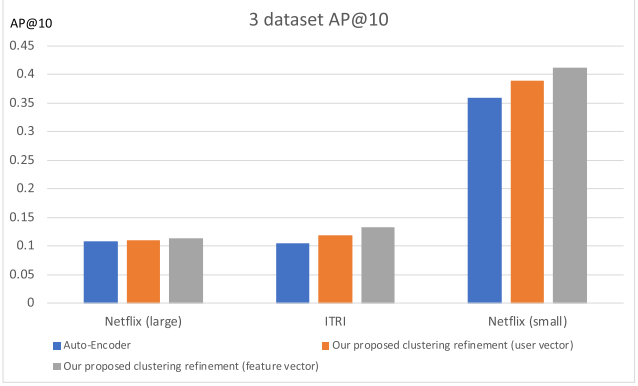

FIGURE 11. The comparison of AP@10 between three models.

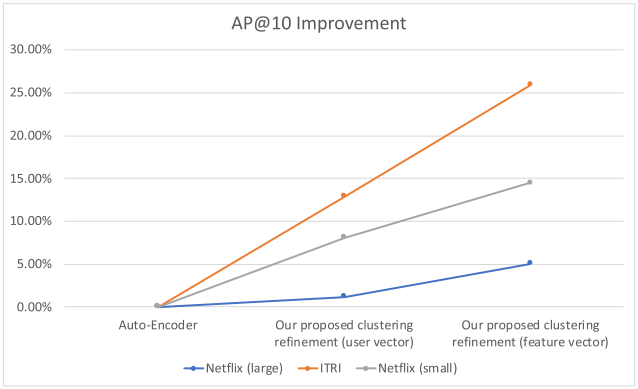

FIGURE 12. Improvements of AP@10 between three models.

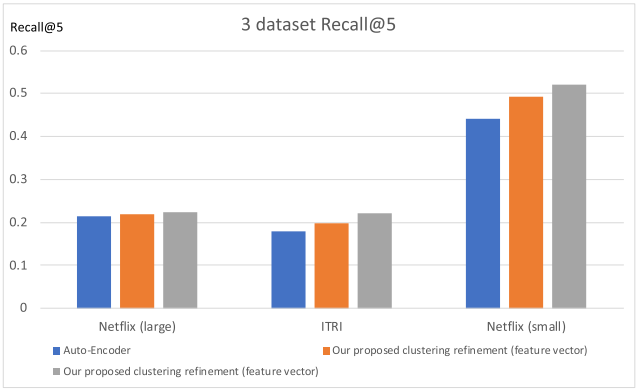

FIGURE 13. The comparison of Recall@5 between three models.

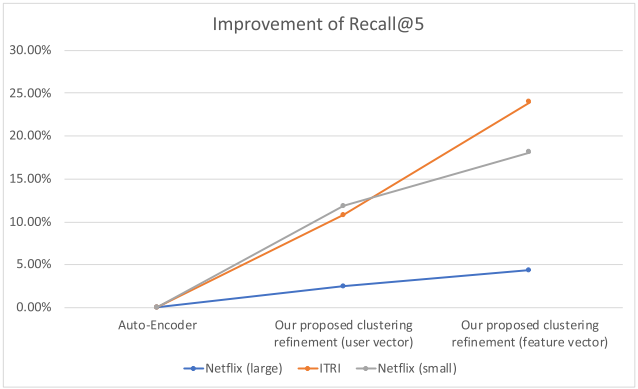

FIGURE 14. Improvements of Recall@5 between three models.

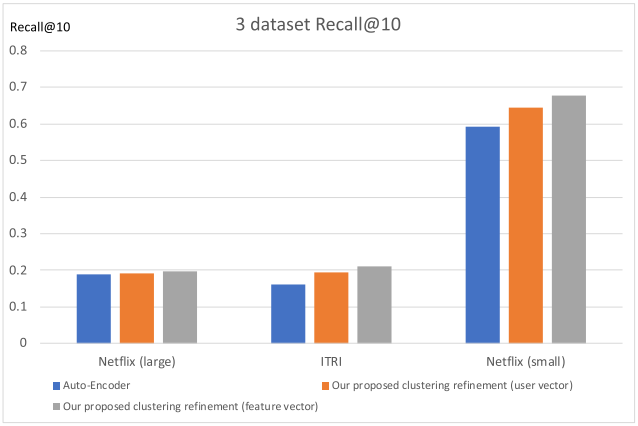

FIGURE 15. The comparison of Recall@10 between three models.

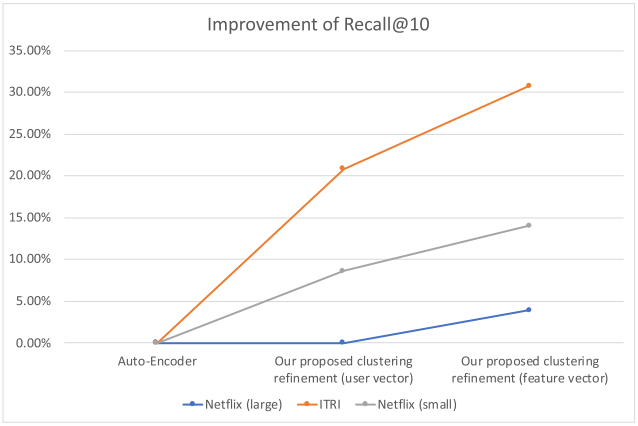

FIGURE 16. Improvements of Recall@10 between three models.

increased by 5.08%. In Fig. 16, by using our proposed model, the improvement can be increased by 30% and 4.37% in ITRI dataset and Netflix (large) dataset, respectively.

D. POTENTIAL TOP-K PREDICTION IN AN UPCOMING TIME PERIOD

1) DATA DESCRIPTION AND INITIAL SETTING

ITRI, Netflix (small) and Netflix (1 Year) datasets are considered in this section. As mentioned in Fig. 2, in Table 4, we choose 1 week, 3 months, and 2 weeks as one time period

TABLE 4. Time period settings.

	Time elapse	Time period	Avg. view count in a time period
ITRI	19/06/2016~14/11/2016	1 week	4,425
Netflix (small)	16/01/2000~31/12/2005	3 months	1,258
Netflix (1 Year)	01/01/2005~31/12/2005	2 weeks	1,232,840

for ITRI dataset, Netflix (small) dataset, and Netflix (1 Year) dataset, respectively. Note that 3 months for Netflix (small) dataset seems long; however, the prediction might be useless if the view count is quite small. Also, to find out the appropriate number of hidden unit that has the best performance, we have tried various lengths of hidden units ranging from 20 to 100 (20, 40, 60, 80, and 100). Table 4 and Table 5 show the parameter settings of three datasets. Table 6 shows the selected hidden unit lengths of model for three datasets.

2) BASELINE MODEL

The baseline model we use as comparisons are 1) the expert-based model in [20] (called expert-based) and

TABLE 5. The initial settings of next time period top-*K*.

	# of users	# of clusters	# of videos	Weight
ITRI	220	10	885	100
Netflix (small)	767	10	74	30
Netflix (1 Y)	60,699	50	8,330	320

TABLE 6. The selected lengths of hidden units in three models.

	ITRI	Netflix (small)	Netflix (1Y)
Clustering	20	20	80
Non-clustering	20	20	100

2) a brute force approach. In [20], they adopted K best experts as forecasters to predict next week video solicitations by calculating each expert's prediction loss. As suggested in [20], in this section, we choose the following experts: Double Exponential Smoothing (DES) expert, Arithmetical Moving Average (AMA) adjusted expert, Dynamic Basic (DB) expert, Constant Basic (CB) expert and Geometrical Moving Average (GMA) adjusted expert. Also, we find that the best K is setting to 2 (i.e., $K = 2$). Notice that the baseline model is a small data analysis, and ours is based on the massive data analysis. The other baseline we use is brute force approach i.e. we directly adopt the Top-K videos in the current time period as the prediction result of Top-K popular videos. This baseline approach is compared in terms of recalls and F1-score of Top-K popular prediction in an upcoming time period in Table 7 and IX.

3) PREDICTION ACCURACY WITHOUT NEW VIDEOS

Table 7 shows the recalls of Top-K of three datasets in an upcoming time period. The numerical results show that both clustering autoencoder and non-clustering autoencoder model outperform the expert-based model. Furthermore, the performance of using clustering is constantly better than that of using non-clustering. It is easily to observe that by using clustering, Netflix (1 Year) ensures the recall values at least 0.78 for predicting Top-30, Top-50, and Top-100. Netflix (small) also approves the recall values around 0.69 and 0.73 for Top-10 and Top-30, respectively when using clustering. However, the recall values are only 0.61, 0.55, and 0.51 in ITRI because, as mentioned before, ITRI data are traces of some pilot runs with a relatively small number of users, and it is harder to predict Top-K eventually.

On the other hand, compared with the other baseline (brute force), the recalls of the brute force approach in Netflix (small) and Netflix (1 Year) are close to the recalls of our refined models. The recalls in Netflix (1 Year) are slightly better than us. We have found the reason is that Netflix (1 Year) data are composed of movies and series and the popularity of movies and series remains fairly static for weeks or months. However, the ITRI dataset is composed of YouTube videos in which the popularity does not stay constant compared to movies and series; thus, the recalls of brute force approach in ITRI is only around 0.22. When comparing to brute force approach, our proposed methods have the ability to capture preferences of users over time and

TABLE 7. Upcoming time period prediction recall.

		Top 5	Top 10	Top 30
Netflix (small)	clustering	0.583333	0.691667	0.730556
	Non-clustering	0.516667	0.646154	0.7179
	Expert-based	0.43333	0.546154	0.630769
	Brute force	0.583333	0.661538	0.723077
ITRI	clustering	0.618182	0.554545	0.515152
	Non-clustering	0.563636	0.518182	0.5
	Expert-based	0.4	0.316667	0.430556
	Brute force	0.200000	0.225000	0.252778
Netflix (1 Year)		Top 30	Top 50	Top 100
	clustering	0.802564	0.783077	0.786154
	Non-clustering	0.772051	0.764615	0.769231
	Expert-based	0.689744	0.72	0.738462
	Brute force	0.810256	0.796923	0.816154

TABLE 8. Upcoming time period prediction still-in recall.

		Top 5	Top 10	Top 30
Netflix (small)	clustering	0.865833	0.88328	0.931299
	Non-clustering	0.7825	0.862937	0.926116
	Expert-based	0.548611	0.781471	0.818634
ITRI	clustering	1	0.915043	0.8406
	Non-clustering	0.954545	0.911255	0.847703
	Expert-based	0.75	0.585218	0.70519
Netflix (1 Year)		Top 30	Top 50	Top 100
	clustering	0.974821	0.951442	0.93148
	Non-clustering	0.926119	0.926857	0.911501
	Expert-based	0.806105	0.879669	0.861168

TABLE 9. F1 score of upcoming time period top-K prediction.

	Top 5	Top 10	Top 30	Top 50	Top 100
Netflix (small) clustering	0.583333	0.691667	0.730556	#	#
Netflix (small) non-clustering	0.516667	0.646154	0.7179	#	#
Netflix (small) Expert-based	0.43333	0.546154	0.630769	#	#
Netflix (small) Brute force	0.583333	0.661538	0.723077	#	#
ITRI clustering	0.618182	0.554545	0.515152	#	#
ITRI non-clustering	0.563636	0.518182	0.5	#	#
ITRI Expert-based	0.4	0.316667	0.430556	#	#
ITRI Brute force	0.200000	0.225000	0.252778	#	#
Netflix (1 Y) clustering	#	#	0.802564	0.783077	0.786154
Netflix (1 Y) non-clustering	#	#	0.771795	0.776923	0.767692
Netflix (1 Y) Expert-based	#	#	0.689744	0.72	0.738462
Netflix (1 Y) Brute force	#	#	0.810256	0.796923	0.816154

have better performance to predict the popular videos in an upcoming time period if the popularity alters more frequently.

The prediction results for all three datasets in an upcoming time period are not very good at first glance, and ITRI dataset even gets only around 0.5 in recall. As mentioned before, the apparent reasoning is the first appearance issue. For example, some videos (such as Avengers) which are very popular will definitely be in the Top-K list when first released. This will be difficult for our prediction model to predict the new coming videos if there is no priori knowledge at all. Therefore, we have to define a fair index, Still_in_Recall@K, to measure the reasonable recall score. Accordingly, in the experimental results, we focus on the videos that are in the Top-K list currently and will be still-in for an upcoming time period Top-K list together.

4) PREDICTION ACCURACY WITH NEW VIDEOS

Table 8 demonstrates the still-in recalls of Top-K list for three datasets. It shows that our proposed model (clustering) performs well in predicting the still-in next time Top-K popular videos. It is noticeable that prediction recalls can achieve 0.9 for all three datasets for some Top-K. Particularly, the prediction recalls (Still_in_Recall@K, $K = 5, 10, 30$) for Netflix (1 Year) dataset are all higher than 0.93 when applying clustering. It also demonstrates that if the video appeared recently, the model would learn the rating information of the

TABLE 10. F1 score of still-in top-*K* prediction.

	Top 5	Top 10	Top 30	Top 50	Top 100
Netflix (small) clustering	0.659061	0.72272	0.8047	#	#
Netflix (small) non-clustering	0.596561	0.703384	0.784637	#	#
Netflix (small) Expert-based	0.410053	0.571221	0.684379	#	#
ITRI clustering	0.625541	0.608304	0.558759	#	#
ITRI non-clustering	0.585137	0.607042	0.563188	#	#
ITRI Expert-based	0.407738	0.358414	0.414835	#	#
Netflix (1 Y) clustering	#	#	0.870565	0.842823	0.836595
Netflix (1 Y) non-clustering	#	#	0.8028	0.8027	0.7976
Netflix (1 Y) Expert-based	#	#	0.720297	0.780357	0.773707

TABLE 11. Prediction recall on various hidden lengths.

Different Hidden Lengths			10	20	40	60	80	100
Netflix (small)	clustering	Top 5	0.533333	**0.583333**	0.523077	0.523077	0.492308	0.507692
		Top 10	0.658333	**0.691667**	0.638462	0.653846	0.630769	0.623077
		Top 30	0.727778	**0.730556**	0.694872	0.712821	0.715385	0.710256
	Non-clustering	Top 5	0.492308	**0.516667**	0.476923	0.492308	0.492308	0.492308
		Top 10	0.638462	**0.646154**	0.653846	0.653846	0.669231	0.669231
		Top 30	0.710256	**0.7179**	0.710256	0.710256	0.715385	0.710513
ITRI	clustering	Top 5	0.545455	**0.618182**	0.563636	0.509091	0.545455	0.490909
		Top 10	0.518182	**0.554545**	0.545455	0.554545	0.509091	0.5
		Top 30	0.506061	**0.515152**	0.478788	0.484848	0.509091	0.481818
	Non-clustering	Top 5	0.506061	**0.563636**	0.55	0.477778	0.533333	0.533333
		Top 10	0.5	**0.518182**	0.491667	0.491667	0.516667	0.508333
		Top 30	0.50303	**0.5**	0.480556	0.533333	0.475	0.483333
Netflix (1 Y)	clustering	Top 30	0.774359	0.774359	0.774359	0.805128	**0.802564**	0.797436
		Top 50	0.773846	0.770769	0.770769	0.784615	**0.783077**	0.789231
		Top 100	0.773077	0.776154	0.776154	0.784615	**0.786154**	0.79
	Non-clustering	Top 30	0.73333	0.753846	0.764103	0.769231	0.771795	**0.771795**
		Top 50	0.749231	0.763077	0.766154	0.769231	0.773846	**0.776923**
		Top 100	0.746923	0.748718	0.76	0.763077	0.765385	**0.767692**

video and would better forecast the temporal dynamics then. Again, this demonstrates that our Top-*K* forecasting method has near-optimal solutions for all these datasets in the sense of still in.

5) F1 SCORE OF PREDICTION ACCURACY WITH OR WITHOUT NEW VIDEOS

F1 score which combines both recall and precision is used to evaluate the overall performance. Table 9 shows the F1 score for prediction of three datasets in an upcoming time period and Table 10 shows the F1 score for still-in Top-*K* prediction of three datasets. Our proposed method (clustering) outperforms other methods in the three datasets with different sizes. Compared to the non-clustering method, the clustering method has better F1 scores (i.e., higher than 0.8) for still-in Top-*K* in the Netflix (1 Year) dataset.

6) PREDICTION RECALL ON HIDDEN LAYERS WITH DIFFERENT LENGTHS

Table 11 shows the prediction recalls of three datasets with various lengths of hidden layer. Numbers marked as red are the best results amongst various lengths. As we can see,

in smaller datasets, the values get worse or equal performance by using a longer hidden vector length. We have found the reason is that in small datasets, the model can learn enough information without too much noise by setting hidden vector as relatively small length (e.g., 20). However, the model needs larger hidden vector lengths (e.g., 80 and 100) for large datasets because there is probably much more useful information which needs to be stored.

V. CONCLUSION AND FUTURE WORK

We propose a method to make use of CDAE in predicting potential Top-K popular videos. At the beginning, we separate data into training data and test data according to separation point and then, we train the autoencoder model based on their previous rating history. We utilize the trained model to predict future potential Top-K popular videos.

In order to improve the prediction accuracy, we propose an autoencoder with clustering refinement, which categorizes user-specific vectors (user-specific vector or feature vector) and refines the original autoencoder model to boost recommendation accuracy. The experimental results demonstrate that our method increases significantly the Average Precision (AP) and the Recall values by nearly 30%.

After making a big progress in improving original collaborative autoencoder model, we use our proposed autoencoder with clustering refinement model to predict potential Top-K popular videos for each time period. We also firstly separate data into training data and test data according to separation point. After training autoencoder model, we cluster users into groups by their user-specific vectors and fine-tune autoencoder model according to each cluster. We, then, utilize our proposed method, mentioned in sub-section III.A.2, to get Top-K popular videos for each cluster. Finally, calculate scores for predicted Top-K popular videos of each cluster by using the ensemble method to aggregate and obtain the overall Top-K popular videos of a time period.

In our experiments, we use ITRI, Netflix (small), Netflix (1 Year), Netflix (large) 4 datasets. For autoencoder refinement with clustering method, the result shows that our proposed method can perform better than the original model in AP@5 by 9.78% (ITRI), 9.98% (Netflix small) and 4.7% (Netflix large). For potential Top-K popular videos prediction in an upcoming time period, the proposed method performs better than original model in Recall by 7.04% (Netflix small), 3.4% (ITRI) and 0.9% (Netflix 1 Year). As shown in Fig. 8, the percentages of new coming videos in Top-K lists for three datasets, it is easily seen that the proposed method is near optimal. The experimental results in Table 7 and 8 demonstrate that our Top-K forecasting method perform near-optimal solutions for various datasets.

In our future work, based on the clustering-based improvement to the autoencoder models [5], [6], firstly, we will apply more complex models [21], [22] to further boost the performance gain to enhance the prediction of recommendations. Secondly, in addition to video rating history, we will apply other environment factors (e.g., sound and picture) to improve prediction accuracy. Thirdly, our proposed model is not capable of dealing with temporal dynamics from a rating history, especially for the first appearance issue. We think that the mixture of content-based approach would give some hints.

REFERENCES

[1] F. Ricci, L. Rokach, and B. Shapira, *Introduction to Recommender Systems Handbook*. Boston, MA, USA: Springer, 2011, pp. 1–35.
[2] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proc. 17th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Diego, CA, USA, 2011, pp. 448–456.
[3] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Proc. 21st Annu. Conf. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Dec. 2007, pp. 1257–1264.
[4] H. Wang, X. Shi, and D.-Y. Yeung, "Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks," in *Proc. 29th Annu. Conf. Neural Inf. Process. Syst.*, Barcelona, Spain, Dec. 2016, pp. 415–423.
[5] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "AutoRec: Autoencoders meet collaborative filtering," in *Proc. 24th Int. Conf. World Wide Web*, Florence, Italy, May 2015, pp. 111–112.
[6] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for Top-N recommender systems," in *Proc. 9th ACM Int. Conf. Web Search Data Mining*, San Francisco, CA, USA, Feb. 2016, pp. 153–162.
[7] *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update*, Cisco Syst., San Jose, CA, USA, 2016.
[8] I. Hwang, B. Song, and S. S. Soliman, "A holistic view on hyper-dense heterogeneous and small cell networks," *IEEE Commun. Mag.*, vol. 51, no. 6, pp. 20–27, Jun. 2013.
[9] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—A key technology towards 5G," *ETSI White Paper*, vol. 11, no. 11, pp. 1–16, 2015.
[10] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web*, Hong Kong, May 2001, pp. 285–295.
[11] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo, "Exploring high-order user preference on the knowledge graph for recommender systems," *ACM Trans. Inf. Syst.*, vol. 37, no. 3, pp. 1–26, Jul. 2019.
[12] Y. Suzuki and T. Ozaki, "Stacked denoising autoencoder-based deep collaborative filtering using the change of similarity," in *Proc. 31st Int. Conf. Adv. Inf. Netw. Appl. Workshops (WAINA)*, Taipei, Taiwan, Mar. 2017, pp. 498–502.
[13] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *Proc. Annu. Int. Conf. Learn. Represent.*, San Diego, CA, USA, May 2015, pp. 1–8.
[14] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, "Recurrent recommender networks," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, Cambridge, U.K., Feb. 2017, pp. 495–503.
[15] F. Xue, X. He, X. Wang, J. Xu, K. Liu, and R. Hong, "Deep item-based collaborative filtering for Top-N recommendation," *ACM Trans. Inf. Syst.*, vol. 37, no. 3, pp. 1–25, Jul. 2019.
[16] W. Lu, F.-L. Chung, W. Jiang, M. Ester, and W. Liu, "A deep Bayesian tensor-based system for video recommendation," *ACM Trans. Inf. Syst.*, vol. 37, no. 1, pp. 1–22, Jan. 2019.
[17] Z. Chen, J. Lee, T. Q. S. Quek, and M. Kountouris, "Cooperative caching and transmission design in cluster-centric small cell networks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 5, pp. 3401–3415, May 2017.
[18] Y.-T. Chen, C.-C. Yen, Y.-T. Lin, and J.-S. Wang, "Cooperative caching plan of popular videos for mobile users by grouping preferences," in *Proc. IEEE 16th Intl Conf Dependable, Autonomic Secure Comput.*, Athens, Greece, Aug. 2018, pp. 762–769.
[19] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proc. 18th Annu. ACM-SIAM Symp. Discrete Algorithms*, Jun. 2007, pp. 1–4.
[20] N. Ben Hassine, D. Marinca, P. Minet, and D. Barth, "Expert-based on-line learning and prediction in content delivery networks," in *Proc. Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Paphos, Cyprus, Sep. 2016, pp. 182–187.
[21] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," in *Proc. Annu. Int. Conf. Learn. Represent.*, San Juan, Puerto Rico, May 2016, pp. 1–7.
[22] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *Proc. Annu. Int. Conf. Learn. Represent.*, Banff, Canada, Apr. 2014, pp. 1–7.

YU-TAI LIN received the B.S. degree in engineering science from National Cheng Kung University, Tainan, Taiwan, in 2016, and the M.S. degree in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 2018. He is currently working at Synology Inc. as a software engineer. He received the Research Creativity Award from the Ministry of Science and Technology for recognizing his research potential in 2016.

CHIA-CHENG YEN received the B.S. degree from Fu Jen Catholic University, New Taipei, Taiwan, in 2012, and the M.S. degree from National Tsing Hua University, Hsinchu, Taiwan, in 2014, all in computer science. He is currently pursuing the Ph.D. degree with the Department of Computer Science, University of California at Davis, CA, USA. He works with the C3PO Research Group. His research interests include reinforcement learning, traffic signal control, cyber-security, and wireless sensor networks.

JIA-SHUNG WANG (Member, IEEE) received the B.S. degree in mathematics from National Taiwan University, Taipei, Taiwan, in 1978, and the M.S. and Ph.D. degrees in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 1983 and 1986, respectively. In 1986, he joined the Department of Computer Science, National Tsing Hua University, as an Associate Professor, where he became a Full Professor, in 1995. His current research interests include several aspects of multimedia networking, video coding, and sensor networks.