Learning to Transfer Graph Embeddings for Inductive Graph based Recommendation

Le Wu
Key Laboratory of Knowledge Engineering with Big Data, Hefei University of Technology
School of Computer Science and Information Engineering, Hefei University of Technology
lewu.ustc@gmail.com

Yonghui Yang
Key Laboratory of Knowledge Engineering with Big Data, Hefei University of Technology
School of Computer Science and Information Engineering, Hefei University of Technology
yyh.hfut@gmail.com

Lei Chen
Key Laboratory of Knowledge Engineering with Big Data, Hefei University of Technology
School of Computer Science and Information Engineering, Hefei University of Technology
chenlei.hfut@gmail.com

Defu Lian
University of Science and Technology of China
liandefu@ustc.edu.cn

Richang Hong
Key Laboratory of Knowledge Engineering with Big Data, Hefei University of Technology
School of Computer Science and Information Engineering, Hefei University of Technology
hongrc.hfut@gmail.com

Meng Wang*
Key Laboratory of Knowledge Engineering with Big Data, Hefei University of Technology
School of Computer Science and Information Engineering, Hefei University of Technology
eric.mengwang@gmail.com

ABSTRACT

With the increasing availability of videos, how to edit them and present the most interesting parts to users, i.e., video highlight, has become an urgent need with many broad applications. As users' visual preferences are subjective and vary from person to person, previous generalized video highlight extraction models fail to tailor to users' unique preferences. In this paper, we study the problem of personalized video highlight recommendation with rich visual content. By dividing each video into non-overlapping segments, we formulate the problem as a personalized segment recommendation task with many new segments in the test stage. The key challenges of this problem lie in: the *cold-start users* with limited video highlight records in the training data and *new segments* without any user ratings at the test stage. To tackle these challenges, an intuitive idea is to formulate a user-item interaction graph and perform inductive graph neural network based models for better user and item embedding learning. However, the graph embedding models fail to generalize to unseen items as these models rely on the item content feature and item link information for item embedding calculation. To this end, we propose an inductive *G*raph based *Trans*fer learning framework for personalized video highlight *Re*commendation (*TransGRec*). TransGRec is composed of two parts: a graph neural network followed by an item embedding transfer network. Specifically, the graph neural network part exploits the higher-order proximity between users and segments to alleviate the user cold-start problem. The transfer network is designed to approximate the learned item embeddings from graph neural networks by taking each item's visual content as input, in order to tackle the new segment problem in the test phase. We design two detailed implementations of the transfer learning optimization function, and we show how the two parts of TransGRec can be efficiently optimized with different transfer learning optimization functions. Please note that, our proposed framework is generally applicable to any inductive graph based recommendation model to address the new node problem without any link structure. Finally, extensive experimental results on a real-world dataset clearly show the effectiveness of our proposed model.

CCS CONCEPTS

• **Information systems** → **Personalization**; **Recommender systems**; *Content ranking*.

KEYWORDS

content based recommendation, inductive graph learning, graph neural network

ACM Reference Format:
Le Wu, Yonghui Yang, Lei Chen, Defu Lian, Richang Hong, and Meng Wang. 2020. Learning to Transfer Graph Embeddings for Inductive Graph based Recommendation . In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401145

*Corresponding Author

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401145

1 INTRODUCTION

With the increasing availability of camera devices, videos are ubiquitous on entertainment and social networking platforms. As these videos are usually unstructured and long-running, it is non-trivial to directly browse the interesting and representative parts and share these parts in the social media. Editing such videos into the highlight segments, i.e., the most interesting or representative parts, and presenting these parts is a natural choice. This real-world demand has raised many practical application scenarios, such as increasing user satisfaction for easy GIF creation, sharing and video preview, and boosting the platform prosperity with video promotion and advertising.

In fact, video highlight extraction is closely related to concepts such as video summarization in the computer vision area, which selects representative parts from videos with well-defined objective functions [23, 30, 47]. After that, the same segments are sent to all users. However, users' visual preferences are not universal but vary from person to person. E.g., with a short video that presents the features of a smartphone, the technical fans like the segments that introduce the detailed parameters of the processor, while others prefer to view the photos taken by the smartphone. As a result, instead of presenting the same highlight parts to all users, an ideal video highlight system should suggest and recommend personalized highlight parts to tailor to users' personalized preferences.

Crucially to the success of personalized video highlight recommendation is obtaining users' historical records of the selected highlight parts for personalization. Luckily, many online highlight creation tools, such as *Gifs.com*, have recorded users' manually selection of their liked video intervals, making it possible for the personalization task. Recently, researchers proposed the problem of personalized highlight recommendation [32]. Intuitively, by dividing each video into a set of non-overlapping segments, the personalized video highlight recommendation task asks: when a user opens a video page, is it possible to automatically suggest (recommend) segments that track her personalized visual preference? In fact, by drawing analogy between an item from an item set, and a segment from a video, it is natural to deem this task as a recommendation problem. In the following, for illustration convenience, we do not distinguish the terms of item and segment.

As new videos emerge from time to time, the video highlight recommendation task is quite challenging as many candidate segments at the test time have never appeared in the training data nor rated by any user. Collaborative Filtering (CF) based recommendation models fail as they rely on the users' historical behaviors to items and could not generalize to unseen items. Therefore, it is a natural choice to design content based models for personalized video highlight recommendation. With the huge success of deep learning models for image and video processing [1, 8], by extracting item semantic features from state-of-the-art deep neural models, most of these models focus on how to align both users and items in a new semantic space by exploiting users' historical behavior data [24, 32, 39]. Some recent works also proposed hybrid recommendation models, where the new item recommendation scenario degenerates to the content based recommendation [40, 41]. All these deep learning based content recommendation models and hybrid models showed better performance to tackle the new item problem. However, in the recommendation process, as each user has very limited historical records, the performances of these models are still far from satisfactory due to the cold-start user problem.

In this paper, we study the problem of personalized video highlight recommendation with two challenges mentioned above: i.e., *cold-start user* problem in the training stage and *new item* without any link structure at the test time. As users naturally form an attributed user-item bipartite graph, an intuitive idea is to perform the graph embedding learning models to model the higher-order graph structure. As such, the correlations of users' rating records can be exploited for better user and item embedding learning, and the cold-start user problem can be partially alleviated. The graph learning process can be seen as learning a mapping function that takes the item visual feature as well as the item's local link structure as input. However, it fails in the test stage as new items are not rated by any user in the test stage, indicating there is no link structure for item embedding learning. Therefore, a natural idea is that: given the initial item content and the item embedding output by the graph neural network in the training stage, can we design an approximation function to mimic the graph embedding output for the new items in the test stage?

To this end, we propose a general framework: an inductive *G*raph based *Trans*fer learning framework for personalized video highlight *Rec*ommendation (TransGRec). TransGRec is composed of two parts: a graph neural network and a transfer item embedding learning network. The graph neural network injects the correlations of users' historical item preferences for better user and item embedding learning, and alleviates the user cold-start problem in the training data. The transfer network aims to approximate the item embeddings from graph neural networks by taking an item's visual content as input, such to tackle the missing link structure for the test items. We design two different optimization functions for the transfer network with different assumptions. We show the two parts of TransGRec can be jointly trained in an efficient way. In fact, our proposed TransGRec framework is generally applicable to inductive graph based recommender models with unseen users or items. Finally, we conduct extensive experimental results on a real-world dataset to show the superiority of our proposed framework.

2 DATASET DESCRIPTION

The dataset is based on a popular personalized video highlight website *Gifs.com*, which is introduced by Molino et al. [32] and publicly available [1]. This website allows users to manually select a video highlight part to create GIFs, which are short and punchy with little space. When a user selects the time intervals of a video she is interested in, her action is recorded as a quadruple: $< u, v, t\_s, t\_e >$, with u denotes the user, v is the video ID and t_s denotes the start time of the highlight and t_e is the end time of the selected highlight part. The original dataset contains about 14,000 users, 119,938 videos and 225,015 annotations.

At the video segment preprocessing step, for each video v, similar as many (personalized) video highlight detection models [32, 47], we start by dividing each video into a set of non-overlapping segments $S^v = \{s_1^v, s_2^v, ..., s_{|v|}^v\}$. This video segmentation could be

[1] https://github.com/gyglim/personalized-highlights-dataset

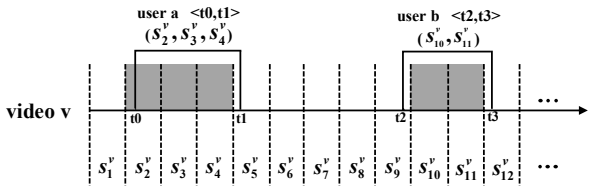

Figure 1: The preprocessing for user-segment records.

implemented by video shot detection based models or simply use the average time split technique [10]. As most video highlights are short, we use a simple video segmentation model that equally split 5 seconds as a segment. After that, for each annotation record of user a to video v, if the overlap percentage between any segment s_i^v ($s_i^v \in S^v$) of this video and her current record is larger than a predefined threshold θ of the segment, it is considered as user a's positive segments, i.e., $r_{ai} = 1$. Without confusion, we would simply replace s_i^v with i to denote a segment. The remaining segments of the video are those that the user has not selected. In this paper, we set the threshold $\theta = 50\%$. We show the preprocessing step of two users in Figure 1. E.g., for user a, her highlight parts span from s_2^v to s_5^v, as the liked parts in s_2^v is larger than θ, while the liked parts in s_5^v is smaller than 50%. Therefore, we set s_2^v, s_3^v, and s_4^v as the three positive segment records of this user. After data segmentation, we have 6,527 users, 5,137 videos with 55,957 segments. Among the 55,957 segments, 25,777 of them have been liked by users, with 41,119 user-segment records. For better illustration, we plot the user distribution of rated videos and rated segments in Figure 2. The distributions roughly follow the power law, with many users have very limited rating records.

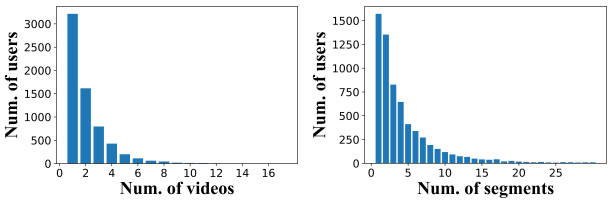

Figure 2: Data distribution: (a)number of rated videos per user; (b)number of rated segments per user.

Let $\mathbf{R}$ denote the user-segment rating matrix, in which r_{ai} is the detailed preference of user a to segment i. The personalized video highlight recommendation task asks: with user-segment rating matrix $\mathbf{R}$, for each user u to each test video v, our goal is to recommend Top-N ranking list of segments which meet each user's personalized preference. .

3 THE PROPOSED MODEL

In this section, we introduce a *G*raph based *Trans*fer learning framework for personalized video highlight *Rec*ommendation (*TransGRec*). After the data preprocessing, in the training process, with the user-segment rating matrix $\mathbf{R}$, we construct a user-item bipartite graph $\mathcal{G} =< \mathcal{U} \cup \mathcal{I}, \mathbf{R} >$, with $\mathcal{U}$ denotes the user set with M users ($|\mathcal{U}| = M$) and $\mathcal{I}$ ($|\mathcal{I}| = N$) is the item (segment) set extracted from all the training data. $\mathbf{R} = [r_{ai}]_{M \times N}$ is the edge set, with $r_{ai} = 1$ denotes user a shows preference for segment i. As videos change quickly in the real world, in the test stage, most test videos are new and have not been rated by users. In other words, these test items neither appear in the training data, nor have been rated by any users.

Given the characteristics of the problem, as shown in Figure 3, there are two key parts of the proposed framework. First, by turning users' historical records into a graph structure, TransGRec represents the user-segment rating behavior as an attributed graph, and uses Graph Neural Networks (GNN) for user and item representation learning. Therefore, the higher-order graph structure is leveraged in the user and item embedding learning process, which could alleviate the cold-start problem. Second, most test videos are new and never appear in the training data, neither do they have any rating records. Under such a situation, graph neural network based inductive learning models fail as each item relies on its content and the local link structure for item embedding learning. To tackle the new item situation without edge in the test stage, we propose a transfer learning model (T) that learns to transform each item's visual input into an approximated embedding in the GNN output space. The goodness of the transfer network is measured by comparing the results of the approximated item embedding, and the real item embedding from the output of the graph neural network in the training data. Therefore, in the test stage, for each item that has not appeared in the training data, the transfer network could be applied to approximate each item's embedding in the inductive setting without any link structure. Then, the predicted rating is learned by comparing the similarity between the user-item pair in the final embedding space.

Given the two parts of the proposed TransGRec structure, the overall loss function is naturally defined as combining the loss functions of the two parts as:

$$\mathcal{L} = \mathcal{L}_{GNN} + \lambda \mathcal{L}_T, \tag{1}$$

where preference prediction loss ($\mathcal{L}_{GNN}$) is the classical rating based optimization loss under GNN modeling, and transfer loss ($\mathcal{L}_T$) denotes the loss for the transfer network T. λ is a balance parameter between these two loss functions.

3.1 Graph Neural Network for Embedding Learning

The graph neural network is mainly composed of four parts: the initial embedding layer, the item embedding fusion layer, the recursive user and item propagation layers, and the prediction layer.

Initial embedding layer. Similar to many embedding based recommendation models, we use $\mathbf{X} \in \mathbb{R}^{D \times M}$ and $\mathbf{Z} \in \mathbb{R}^{D \times N}$ to denote the free embedding matrices of users and items. For each user, as we do not have her profile or related metadata, each user a's initial embedding is denoted as $\mathbf{x}_a$, which is the a-th column of the user free embedding matrix $\mathbf{X} \in \mathbb{R}^{D \times M}$. For each item i, its free embedding is the i-th column of the item free embedding matrix $\mathbf{Z} \in \mathbb{R}^{D \times N}$, i.e., $\mathbf{z}_i$.

Item embedding fusion layer. The item embedding fusion layer fuses the free item embedding and its visual embedding. For each segment that appears in a video, we leverage video representation models for item semantic embedding. Given each item of a

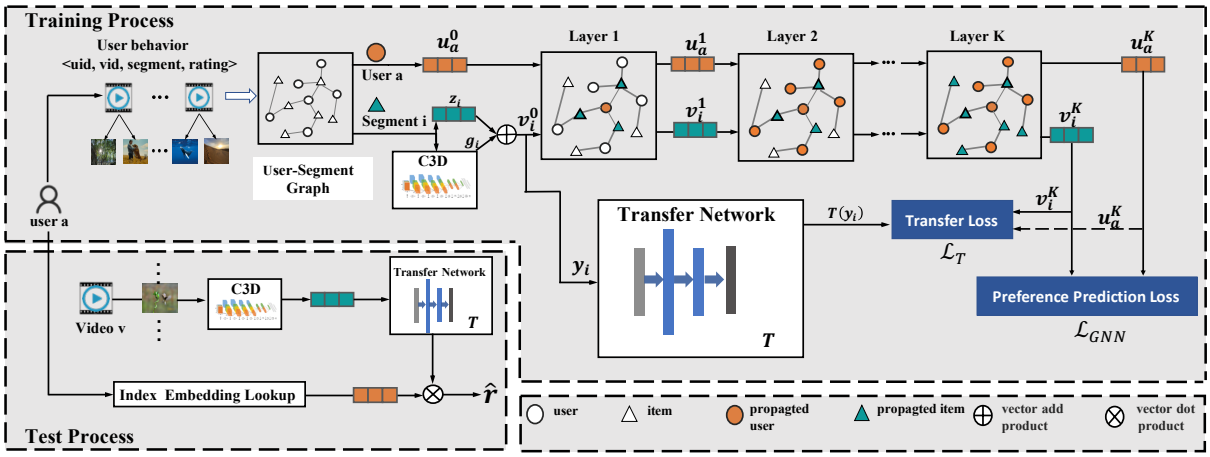

Figure 3: The overall framework of TransGRec.

video segment, similar to many popular visual-based video processing approaches [8, 17], we select Convolutional 3D Networks (C3D) pretrained on the Sports-1M [18] dataset as the video feature extractor. Specifically, we use the output of first connected layer (fc1) of C3D as the visual feature $\mathbf{f}_i \in \mathbb{R}^{4096\times 1}$ of each item i. Then, we use a dimension reduction matrix $\mathbf{W}^0 \in \mathbb{R}^{D\times 4096}$ to reduce the original visual embedding into a low dimension space as:

$$\mathbf{g}_i = \mathbf{W}^0\mathbf{f}_i. \tag{2}$$

Then, for each item i, we could get its fused item embedding as a combination of the free embedding and the visual content embeddding:

$$\mathbf{y}_i = \mathbf{g}_i + \mathbf{z}_i = \mathbf{W}^0\mathbf{f}_i + \mathbf{z}_i. \tag{3}$$

Please note that, there are different kinds of methods that fuse the two parts of item representations, such as concatenation or addition. In this paper, we use the addition operation, as we find the addition operation is more stable in the experiments, and usually achieves better performance.

Embedding propagation layers. This part stacks multiple layers to propagate user and item embeddings of the graph, such that the higher-order graph structure is modeled to refine user and item embeddings. Let $\mathbf{u}_a^k$ denote user a's embedding and $\mathbf{v}_i^k$ as item i's embedding at the k-th propagation layer. As the output of the initial embedding layer is directly sent to the propagation layers, we have $\mathbf{u}_a^0 = \mathbf{x}_a$ and $\mathbf{v}_i^0 = \mathbf{y}_i$ with $k = 0$.

By iteratively feeding the output of all the nodes' embeddings in the graph from k-th propagation layer, each user a's updated embedding $\mathbf{u}_a^{(k+1)}$ at $(k+1)$-th is composed of two steps: a pooling operation that aggregates all the connected neighbors' embeddings at k-th layer into a fixed-length vector representation $\mathbf{u}_{Ra}^{k+1}$, followed by an update step that combines the neighbors' representations and her own embedding at k-th layer. Mathematically, these two steps could be defined as:

$$\mathbf{u}_{Ra}^{k+1} = Pool(\mathbf{v}_j^k | j \in R_a), \tag{4}$$

$$\mathbf{u}_a^{k+1} = \sigma(\mathbf{W}_u^{k+1} \times (\mathbf{u}_a^k + \mathbf{u}_{Ra}^{k+1})), \tag{5}$$

where $R_a = \{i|R_{ai}=1\}$, $R_a \subseteq V$ is the item set that user a interacts with. $\mathbf{W}_u^{k+1} \in \mathbb{R}^{D\times D}$ is the transformation matrix in $(k+1)$-th layer, and $\sigma(x)$ is an activation function. In Eq. (4), the pooling operation in the item neighbor aggregation step is quite flexible, which could be defined as an average pooling that takes the mean of all item neighbors' embedding vectors, or the max pooling that selects the maximum value in each dimension from the embeddings of all the item neighbor set. In the update step of Eq.(5), the pooling vector is first added with the user representation in the previous layer, and then a transformation to get the updated user embedding at $(k+1)$-th layer. Please note that, we have also tried the concatenation operation, and find the addition also performs better.

Given the user embedding process, similarly, let $R_i = \{a|R_{ai} = 1\}$, $R_i \subseteq U$ denote the user set that shows preferences to item i, we could update each item i's embedding $\mathbf{v}_i^k$ at the k-th layer to $\mathbf{v}_i^{(k+1)}$ at $(k+1)$-th layer as:

$$\mathbf{v}_{Ri}^{k+1} = Pool(\mathbf{u}_a^k | a \in R_i), \tag{6}$$

$$\mathbf{v}_i^{k+1} = \sigma(\mathbf{W}_v^{k+1} \times (\mathbf{v}_i^k + \mathbf{v}_{Ri}^{k+1})), \tag{7}$$

Prediction layer. After the embedding propagation layers reaches a defined depth K, we obtain the user embedding $\mathbf{u}_a^K$ and the item embedding $\mathbf{v}_i^K$ at the K-th layer for final user and item embedding:

$$\mathbf{u}_a = \mathbf{u}_a^K, \tag{8}$$

$$\mathbf{v}_i = \mathbf{v}_i^K. \tag{9}$$

Then, we could predict each user a's rating to item i as the inner product between the user embedding $\mathbf{u}_a$ and item embedding $\mathbf{v}_i$:

$$\hat{r}_{ai} = \mathbf{u}_a^T\mathbf{v}_i. \tag{10}$$

Please note that, in the graph neural network part for embedding learning, distinct from classical inductive graph embedding models that take item content as input, we additional add the free item embedding into embedding propagation layers. Adding the free item embedding could allow the graph neural network to learn the collaborative information that is not reflected in the content, which usually shows better performance for many recommendation tasks. Therefore, the final item embedding output from the graph neural

network is a hybrid item representation. Naturally, the above graph neural network part is not inductive and could not generalize to unseen items in the test stage. We would illustrate how to achieve the inductive ability by transfer network introduced in the following subsection.

3.2 Transfer Network for Inductive Item Embedding Learning

As new videos evolve from time to time, in the test stage, most candidate items (segments) have never appeared in the training data. More importantly, most of these new items have not been rated by any users, i.e., most test items are isolated nodes and are not connected to any user in the training data. As each new item's free embedding is not available, as shown in Eq.(3) and Eq.(6), the item embedding fusion layer and the propagation layers fail. To tackle the new item in the test stage, a natural idea is to directly send each candidate item i's content features into a transfer network, and outputs its final embedding $\mathbf{v}_i$ with the learned parameters. Under such setting, we design a transfer network T to transform each item i's embedding in the content space $\mathbf{f}_i$ to approximate its embedding in the final embedding space $\mathbf{v}_i$.

In fact, the mapping function from each item's visual input representation to its final input learned from GNN is complex. To tackle the complex transformation, we choose a Multi-Layer Perception (MLP) to implement the transfer network, as MLPs are demonstrated to be able to approximate any complex function [15]. Therefore, given all pairs of $[\mathbf{f}_i, \mathbf{v}_i]_{i=1}^N$ in the training data as labeled data, the transfer network T with a MLP learns to approximate the output embeding as:

$$\hat{\mathbf{v}}_i = T(y_i) = h^L(...h^1(\mathbf{y}_i)), \tag{11}$$

where $h^l(x) = f(\mathbf{P}^l h^{(l-1)}(x))$ is a function that takes the output of the $(l-1)$-th layer as input. $\mathbf{P}^l$ is the parameter at the l-th layer. Note that as each item's collaborative embedding should be consistent with the range of the output in the GNN, we set $f(x)$ same as σ of Eq. (7).

Euclidean distance based loss. A naive idea of defining the transfer network T is to compare the learned transferred embeddigns with output embeddings from GNN in the Euclidean space as:

$$\mathcal{L}_1 = \sum_{i=1}^{I} ||\hat{\mathbf{v}}_i - \mathbf{v}_i||_F^2, \tag{12}$$

In the above transfer network T, let $\Theta_T = [\mathbf{P}^l]_{k=1}^L$ denote the parameter set. The above Euclidean space assumes that conditional likelihood of the approximated item embedding matrix $\hat{\mathbf{V}}$ learned from the transfer network T follows a Gaussian distribution as: $p(\hat{\mathbf{V}}|\mathbf{V}) \sim \mathcal{N}(mean, \sigma)$, with the mean value is the item embedding learned from the graph neural network GNN. As such, maximizing the log likelihood of the approximated item embedding is equivalent to minimizing the Euclidean distance as shown in Eq.(12).

Adversarial loss. The above Euclidean distance (Eq. 12) fails when $p(\hat{\mathbf{V}}|\mathbf{V})$ is complex, e.g., this distribution has multiple peak points. Simply reducing this likelihood function to a Guassian distribution would fail. Therefore, in order to model the complex conditional distribution, we propose to introduce an adversarial loss based on Generative Adversarial Networks [9]. Besides the transfer network T that generates (fake) approximated item embeddings, a discriminator D parameterized by Θ_D is introduced to distinguish whether the item embedding is real or fake. Specifically, the "real" labels are assigned to the item embeddings that are learned from the output of GNN, while "fake" samples are those approximated item embeddings learned by the transfer network T. The adversarial loss is defined to let the discriminator D and the transfer network T to play the following two-player minimax game as:

$$\arg\max_{\theta_T}\min_{\theta_D} \mathcal{L}_2 = -\sum_{i=1}^{N}\sum_{u=1}^{M}[logD(\mathbf{v}_i, \mathbf{u}_a, r_{ai}) + log(1 - D(T(\mathbf{y}_i), \mathbf{u}_a, r_{ai}))], \tag{13}$$

In the above optimization function, we also feed $\mathbf{u}_a$ and r_{ai} into the discriminator, as introducing these conditional or latent information would make the adversarial training process easier [2, 7]. Since T is implemented with MLPs in Eq.(11), we also use another MLPs to constitute the discriminator D, with $\Theta_D = [\mathbf{Q}^l]_{l=1}^L$ is the parameter set in this discriminator.

3.3 Model Optimization

As the implicit preference is more common in real-world recommender systems, without loss of generality, we use Bayesian Personalized Ranking (BPR) as the preference loss function in GNN [34]:

$$\min_{\Theta_{GNN}} \mathcal{L}_{GNN} = \sum_{a=1}^{M}\sum_{(i,j)\in D_a} -\ln s(\hat{r}_{ai} - \hat{r}_{aj}) + \lambda(||\mathbf{X}||^2 + ||\mathbf{Z}||^2), \tag{14}$$

where $s(x)$ is a sigmoid function, $\Theta_{GNN} = [\mathbf{X}, \mathbf{Z}, \Theta_2 = [\mathbf{W}^k]_{k=0}^K]]$ is the parameter set in the graph neural network, and λ is a regularization parameter. $D_a = \{(i,j)|i \in R_a \wedge j \notin R_a\}$ denotes the pairwise training data for a, with R_a represents the item set that a positively shows feedback and j belongs to the negative itemset. For each user of a positive segment i, it is associated with a corresponding video. Therefore, all the segments that have not been rated by the user are considered as candidate negative items.

Overall optimization function. Given the overall optimization function in Eq.(1), with the detailed GNN based loss (Eq.(14)) and two detailed optimization functions in T, we can get two kinds of overall optimization functions.

1) Overall optimization function with Euclidean loss. For convenience, We call this proposed model as TransGRec-E (Euclidean). By combining the Euclidean loss in Eq.(12), the overall loss function is defined as:

$$\arg\min_{[\Theta_T, \Theta_{GNN}]} \mathcal{L} = \sum_{a=1}^{M}\sum_{(i,j)\in D_a} -\ln s(\hat{r}_{ai} - \hat{r}_{aj}) + \lambda(||\mathbf{X}||^2 + ||\mathbf{Z}||^2) + \sum_{i=1}^{I}||T(\mathbf{y}_i) - \mathbf{v}_i||_F^2 \tag{15}$$

As all parameters in the optimization function are differentiable, we could use gradient descent algorithms to optimization.

2) Overall optimization function with adversarial loss. For convenience, we call this proposed model as TransGRec-A (Adversarial). With the adversarial loss in Eq.(13), there are three sets of

parameters as: $\Theta = [\Theta_{GNN}, \Theta_T, \Theta_D]$. Since the overall optimization function involves both maximization and minimization with regard to different parameter sets. We use alternating update step as:

- Fix Θ_{GNN} and Θ_D, update Θ_T as:

$$\arg\min_{\Theta_T} \sum_{i=1}^{N} \sum_{a=1}^{M} [log(1 - D(T(\mathbf{y}_i)), \mathbf{u}_a, r_{ai})]. \tag{16}$$

- Fix Θ_{GNN} and Θ_T, update Θ_D as:

$$\arg\min_{\Theta_D} \sum_{i=1}^{N} \sum_{a=1}^{M} -[logD(\mathbf{v}_i, \mathbf{u}_a, r_{ai}) + log(1 - D(T(\mathbf{y}_i)), \mathbf{u}_a, r_{ai})]. \tag{17}$$

In practice, we alternate the process of updating Θ_{GNN}, Θ_T and Θ_D, and stops when the loss function converges.

After the training process is finished, the personalized video recommendation process is very easy. The overall flowchart of the test phase is shown at the bottom part of Figure 3. For each user a, we can index the user's final embedding $\mathbf{u}_a$ from the learned user embedding matrix U. For each candidate test segment i, we first take the item feature $\mathbf{f}_i$ as input and get the low dimension visual embedding $\mathbf{g}_i$. Then we can get the approximated item embedding from the transfer network as: $\hat{\mathbf{v}}_i = T(\mathbf{g}_i)$. In summary, the predicted preference of each user-item pair (a, i) can be calculated as:

$$\hat{r}_{ai} = u_a^T \hat{\mathbf{v}}_i \tag{18}$$

3.4 Discussion

Connections with related learning models. TransGRec framework is a concrete application of transfer learning [33]. We formulate the video highlight recommendation problem as an inductive representation learning on graph neural networks. In order to generalize to unseen nodes in test stage, we transfer the embeddings learned in the source network (i.e., GNN) of the training data to learn an approximated item embedding in the target network. In the review based recommendation problem, some researcheres have applied the transfer network to approximate each unseen user-item pair review with the available information in the training data [6, 37]. We differ greatly from these works from the problem definition, the proposed model and the application. These review based recommendation models are transductive learning problems with missing values in the test stage, and do not rely on the graph neural network based approaches. By using MLPs to approximate the graph neural network structure, our work also seems to be correlated with knowledge distillation [14, 28]. The graph neural network can be regarded as the complex teacher network, while the transfer network resembles a student network that distills the knowledge learned from the teacher network.

Generalization of the proposed model. In fact, TransGRec could be seen as an inductive graph based hybrid recommendation framework. With user-item bipartite graph, the item embedding is fused by the free collaborative embedding and the content embedding extracted from video representation learning models. Therefore, the graph neural network could better refine user and item embeddings by preserving the higher-order local structure of this user with recursive feature propagation in this graph. In such a way, we could better represent users and items to alleviate the data sparsity issue. The transfer network learns to transfer the available content representation to the final graph embedding space. Therefore, the proposed framework is generally applicable to many inductive recommendation scenarios with new multimedia items or cold-start users. E.g., in the news recommendation scenario, news articles are highly time-sensitive with many news come from time to time, our proposed framework could well utilize both the article content and the collaborative signals in users' previous behaviors for better user and news representation learning.

Table 1: The statistics of the dataset.

Users	Segments	Segments in the graph
6,527	55,957	25,777
Training records	Validation records	Test records
34,563	3,840	2,716

4 EXPERIMENTS

4.1 Experimental Settings

Experimental setup. As each highlight record is associated with a detailed time, with the preprocessed dataset introduced before, for each user that has at least five video highlight records, we select the last highlight video of each user for model evaluation, which leads to 2716 user-segment records of the test data. Then, we randomly select 10% from the remaining data as the validation set for model tuning. We show the data statistics of the dataset in Table 1.

In model evaluation stage, for each user-segment record, it is associated with a video. We use the unobserved segments of the corresponding video that have not been selected by the user as candidate negative items for recommendation. We use two kinds of metrics from the video highlight domain and the recommendation domain for evaluation. First, as the personalized video highlight recommendation is correlated to video highlight extraction, we adopt two widely used video detection metrics: Mean Average Precision (MAP) and Normalized Meaningful Summary Duration (NMSD) to evaluate the prediction performance on the video level [11, 32]. Specifically, MAP describes the average precision of the ranking of all segments of a video, and the larger value means the better performance. NMSD rates how much of the segments have been watched before the majority of the ground truth segments were shown, so the smaller value means the better performance. Besides, our task is also a Top-N recommendation task, so we use three popular ranking metrics for evaluation from different perspectives: Hit Ratio (HR), Recall [35], and Normalized Discounted Cumulative Gain (NDCG) [36, 44]. HR measures the number of items in the Top-N ranking list that the user likes. Recall measures the number of items that the user likes in the test data that has been successfully predicted in the Top-N ranking list. And NDCG considers the hit positions of the items and gives a higher scores for the hit items in the topper positions. .

Baselines. We compare our proposed model with the following baselines: Video2GIF [11], SVM-D [31], PHD-GIFs [32], DropoutNet [40], CDL [24], and LapReg [12]. Specifically, Video2GIF is a state-of-the-art model for general video highlight detection, which exploits the different visual representations of positive items and

Table 2: Overall performance comparison ↑ means the larger value, the better performance; ↓ means the smaller value, the better performance).

Models	MAP↑	NMSD↓	HR@5↑	NDCG@5↑	Recall@5↑
Video2GIF	0.2075	0.4288	0.1993	0.1651	0.1798
SVM-D	0.2185	0.4180	0.2191	0.1772	0.1991
PHD-GIFs	0.2170	0.4419	0.2228	0.1781	0.2028
DropoutNet	0.2604	0.3886	0.2569	0.2162	0.2353
CDL	0.2706	0.3806	0.2729	0.2304	0.2540
LapReg	0.2828	0.3905	0.2794	0.2360	0.2598
TransGRec-E	0.3113	**0.3702**	0.3066	0.2687	0.2873
TransGRec-A	**0.3174**	0.3769	**0.3153**	**0.2779**	**0.2940**

negative items for modeling. SVM-D and PHD-GIFs are two popular models designed for content based recommendation, with the user embedding is learned from their interacted items. DropoutNet and CDL are content enhanced model that could tackle the new item problem. Specifically, DropoutNet deals with the new user and new item with user dropout and item dropout in the training process [40]. CDL learns the free user content embedding, and the item content embedding in a low latent space given the available training data [24]. LapReg is designed for semi-supervised tasks, with the prediction function similar to the supervised learning models, and it has an additional Laplacian regularization term to capture the correlations of predictions in the graph [4]. In practice, for fair comparison of LapReg, we choose the best prediction model in the remaining baselines as the prediction function in LapReg, and the Laplacian regularization is performed in the user-item bipartite graph. Therefore, these baselines include classical supervised models, deep learning based supervised models, content enhanced recommendation models that can tackle the new item problem, and the graph based semi-supervised models. Please note that, we do not introduce any inductive graph neural network based recommendation models for recommendation, as these models need the unseen nodes to have several links. However, in the video highlight recommendation, most new segments have never been rated by any user. Therefore, these inductive graph neural network based models fail.

Parameter setting. For baselines of Video2GIF and PHD-GIFs, we use the same settings as the original papers with two hidden layer dimensions of 512 and 128 [11, 32]. For all embedding based models (SVM-D, CDL, DropoutNet, TransGRec), we initialize the embedding matrix with a Gaussian distribution with a mean of 0 and variance of 0.01. In the model training process (Eq.(14)), as the unobserved possible negative samples are much more than positive samples, similar to many implicit feedback based recommendation models [3, 34], we use negative sampling technique at each training iteration. Specifically, for each positive pair of user a to item i that comes from video v, we choose the negative samples as follows: we select 10 negative segments form all segments of training data. All segments contain positively selected by other users and unobserved segments of the training video. This negative sampling technique encourages more negative segments appeared in user-item graph with user ratings. The reason is that, if a segment is never rated by any user, it is an isolated point of this graph without any neighbor, and could be learned with higher-order graph structure. In the model learning process, we use Adam optimizer with a learning rate of 0.001 and a mini-batch size of 50. In our TransGRec framework, we choose the embedding dimension D in the set [32, 64, 128, 256], and find $D = 128$ reaches the best performance. Besides, the regularization parameter λ is set in range $[0.1, 1, 10]$, and $\lambda = 1$ reaches the best performance and are is stable. For each propagation layer as shown in Eq.(5) and Eq.(7), we use ReLU function as the activation function to transform the output embedding, and use the mean pooling as the pooling operation. For transfer network T, we choose to use a two-layer MLP. For discriminator (D) in transfer network T, we train a two-layer MLP. There are some parameters in the baselines, we tune all these parameters to ensure these baselines reach the best performance.

4.2 Overall Comparison

We show the evaluation results of various metrics in Table 2. Among all the baselines, Video2GIF is a generalized video highlight recommendation model that presents the same video highlights to all users. All the remaining models show better performance than Video2GIF, indicating the soundness of personalization. DropoutNet and CDL improve over PHD-GIFs and SVD-M about 20% with more precise user embedding. CDL further improves DropoutNet. We guess a possible reason is that, in order to adapt the problem to the new user setting, DropoutNet optimizes the original rating based loss and the new user dropout loss at the same time. Thus, it does not show the best performance under the scenario without new users. As LapReg is implemented with an additional Laplacian regularization term on top of the remaining best baseline (i.e., CDL), LapReg shows better performance compared to CDL on most evaluation metrics.

Our proposed two detailed models of the TransGRec framework consistently outperform the best baselines, showing the effectiveness of better user modeling through user embedding propagation layers in the user-item graph. On average, TransGRec-A improves the best baseline about 13% in MAP, 1% in NMSD, 16% in HR@5, 20% in NDCG@5 and 24% in Recall@5. When comparing TransGRec-A and TransGRec-E, we find on average TransGRec-A shows better performance than TransGRec-E. We guess a possible reason is that TransGRec-A could measure the difference of two complex distributions in non-Euclidean space.

In order to better evaluate the performance of recommendation task, we report the Top-N ranking performance with different N values about various models. The detailed experiment results are shown in Table 3. We also find the similar observations as Table 2, with our proposed TransGRec always shows the best performance under various Top-N values. Based on the overall experimental results, we could empirically conclude that our proposed TransGRec framework outperforms all the baselines under both video detection metrics and ranking metrics.

4.3 Performance under Different Data Sparsity

We split users into different groups and observe the performance under different sparsity. Specifically, we split all users into three groups based on the number of the rated items in the training data, and the performance of all models under different sparsity are shown in Figure 4. Due to page limit, we do not show the results

Table 3: Recommendation metrics comparisons of different Top-N values.

Models	HR@N ↑					Recall@N ↑					NDCG@N↑				
	N=1	N=2	N=3	N=4	N=5	N=1	N=2	N=3	N=4	N=5	N=1	N=2	N=3	N=4	N=5
Video2GIF	0.1477	0.1319	0.1445	0.1702	0.1993	0.0498	0.0795	0.1112	0.1465	0.1798	0.1477	0.1348	0.1398	0.1521	0.1651
SVM-D	0.1371	0.1340	0.1656	0.1934	0.2191	0.0513	0.0863	0.1311	0.1685	0.1991	0.1371	0.1332	0.1505	0.1640	0.1772
PHD-GIFs	0.1435	0.1445	0.1586	0.1966	0.2228	0.0478	0.0888	0.1246	0.1723	0.2028	0.1435	0.1428	0.1480	0.1661	0.1781
DropoutNet	0.1751	0.1804	0.2057	0.2249	0.2569	0.0617	0.1191	0.1641	0.1958	0.2353	0.1751	0.1774	0.1920	0.2015	0.2162
CDL	0.1857	0.1941	0.2278	0.2514	0.2729	0.0714	0.1340	0.1890	0.2254	0.2540	0.1857	0.1901	0.2086	0.2208	0.2304
LapReg	0.1793	0.1962	0.2215	0.2472	0.2794	0.0746	0.1399	0.1833	0.2203	0.2598	0.1793	0.1912	0.2072	0.2208	0.2360
TransGRec-E	0.2215	0.2373	0.2595	0.2815	0.3066	0.0948	0.1712	0.2171	0.2548	0.2873	0.2215	0.2323	0.2453	0.2563	0.2687
TransGRec-A	**0.2363**	**0.2416**	**0.2634**	**0.2915**	**0.3153**	**0.1045**	**0.1767**	**0.2220**	**0.2620**	**0.2940**	**0.2363**	**0.2395**	**0.2519**	**0.2668**	**0.2779**

Table 4: Effects of different propagation layer depth K in TransGRec-E.

Depth K	MAP↑	Improve	NMSD↓	Improve	HR@5↑	Improve	NDCG@5↑	Improve	Recall@5↑	Improve
K=2	**0.3113**	-	**0.3702**	-	**0.3066**	-	**0.2687**	-	**0.2873**	-
K=0	0.3020	-2.99%	0.3921	-5.92%	0.2834	-7.57%	0.2542	-5.40%	0.2617	-8.91%
K=1	0.3073	-1.28%	0.3816	-3.08%	0.2999	-2.19%	0.2617	-2.61%	0.2802	-2.47%
K=3	0.2990	-3.95%	0.3989	-7.75%	0.2941	-4.08%	0.2553	-4.99%	0.2741	-4.59%

Table 5: Effects of different propagation layer depth K in TransGRec-A.

Depth K	MAP↑	Improve	NMSD↓	Improve	HR@5↑	Improve	NDCG@5↑	Improve	Recall@5↑	Improve
K=1	**0.3174**	-	**0.3769**	-	**0.3153**	-	**0.2779**	-	**0.2940**	-
K=0	0.3030	-4.54%	0.3989	-5.84%	0.2861	-9.26%	0.2586	-6.94%	0.2633	-10.44%
K=2	0.3091	-2.61%	0.3854	-2.26%	0.2911	-7.68%	0.2612	-6.01%	0.2723	-7.38%

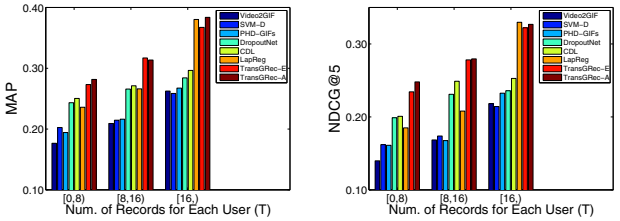

Figure 4: Performance under different data sparsity.

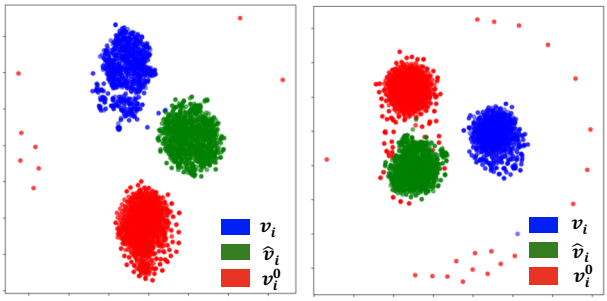

Figure 5: The t-SNE visualization of three types of item embeddings of different models. Color of nodes indicates the type of node embeddings. Red: " the layer-0 embedding in embedding propagation layers (v_i^0)", Blue:"final item embedding output by the embedding propagation layers (v_i)", green:"approximated item embedding learned by the transfer network ($\hat{v}_i$)". The left figure shows the results of TransGRec-E, and the right figure shows the results of TranGRec-A.

of NMSD as it shares similar trends as MAP. Also, the results of HR@5 and Recall@5 are similar to NDCG@5. E.g, for each user in the group of [8, 16), the number of her training records satisfies $8 \leq |\mathbf{R}_a| < 16$. As illustrated in this figure, with the increase of the user interaction records, the performance increases quickly for all models, as all models need to rely on user rated items for embedding learning. Our proposed models consistently outperform the baselines under all user groups. E.g., TransGRec-A improves over the best baselines about 23% on the [0, 8) user group with the NDCG@5 metric. All baselines show similar performance trend, expect the LapReg model, which shows similar performance as TransGRec on the [16,) group. However, LapReg does not perform well on the sparser user groups. We guess a possible reason is that, LapReg relies on the local link structure for graph regularization. When users have limited links, the regularization term is not reliable with limited first-order neighbors. As users have more rating records, the performance of LapReg increases. In contrast, TransGRec could better model the higher-order graph structure, and it shows relatively higher performance even with cold-start users.

4.4 Detailed Model Analysis

In this part, we analyze the impact of different propagation layer in depth K. Table 4 and Table 5 summarize the results of two models with different K values. The column of "Improve" shows the performance changes compared to the best setting of model, i.e., K=2. When the propagation layer depth K=0, TransGRec only uses initialized user embedding and not aggregates neighbor feature. As shown in Table 4, when the propagation depth increases from K=1 to K=2, the performance improves and K=2 reaches the best performance. However, the performance drops when K=3 as three layers may introduce noisy neighbors of the graph. The performance drops when K=3 has also been observed in GCN based models [38, 45].

The same phenomenon also occurs in Table 5, the difference is that the best performance reaches when K=1 for TransGRec-A.

A key characteristic of our proposed TransGRec module is designing a transfer network to approximate the learned embeddings from graph neural networks. Therefore, it is natural to ask: does the transfer network show ability to mimic the graph network embeddings. To answer this question, for both methods of TransGRec-E and TransGRec-A, after the training process converges, we visualize three kinds of item embeddings: the layer-0 embedding in the embedding propagation layers ($\mathbf{v}_i^0$) of the graph neural network, the final item embedding output by the embedding propagation layers ($\mathbf{v}_i$), and the approximated item embedding learned by the transfer network ($\hat{\mathbf{v}}_i$). We randomly select 1000 items and map these three categories of item embeddings into a 2-D space with the t-SNE package [29] . The results are shown in Figure 5. Visualizations of the two models show similar phenomenon. The representation exhibits clustering effects of different kinds of item embeddings. Specifically, the distance between the fused item embedding and graph embedding output is large, showing that the graph neural network could transform the fused item embedding into a different space through iterative graph embedding propagation layers. We observe the relative distance between the transferred embeddings and the initial embeddings, is smaller than that between the the graph output embeddings and the initial embeddings. This phenomenon shows that the transfer network could partially mimic the graph embedding function to learn useful transfer embedding, and validates the effectiveness of the transfer network. However, there are still a gap between the transferred approximated embeddings and the graph embedding output. We guess the reason is that, the graph structure is very complex, while relying on a transfer network could not well capture all the information hidden in the graph. We would leave the problem of how to better design a transfer network that well captures the graph structure as our future work.

5 RELATED WORKS

5.1 Video Highlight and Personalization

Automatic video highlight extraction and summarization deals with selecting representative segments from videos [5, 30]. These models learned interesting, representative, or diversified visual content segments based on well-defined optimization goals [5, 49]. As many users edited videos in online platforms, some researchers proposed to leverage the wisdom of the crowds as ground-truth or priors to guide video highlight extraction [11, 19, 30]. However, these proposed models neglected users' personalized preferences. With the huge boom of video editing platforms and APPs, recently researchers published a personalized video highlight dataset that records each user's liked segments of videos, and proposed a personalized highlight detection model [32]. The personalization is achieved by adapting the input of each user as an aggregation of her liked segments of videos, and the proposed personalization model showed better performance compared to the state-of-the-art generalized highlight extraction model [11], indicating the soundness of personalization in video highlight recommendation. However, as each user's annotated records are limited, the recommendation performance is still far from satisfaction.

5.2 Classical Recommendation Models

Given user-item interaction behavior, CF achieves high performance by learning users and items in a low dimensional collaborative space [26, 27, 34]. However, CF models fail to tackle the new item problem as these models relied on the user-item interaction behavior for recommendation. With the huge success of deep learning in computer vision and related areas, many state-of-the-art content based models learn to align users and items in a semantic content space with deep learning techniques [16, 22, 24]. Some hybrid recommendation models were proposed to leverage both user behavior data and item content for recommendation [13, 40, 41, 43]. However, most of these models could not tackle the new item problem, as each item representation is a hybrid representation with both the collaborative information and the semantic content representation [13]. There are two related hybrid recommendation works that could adapt to new item issue [40, 41]. In the proposed Collaborative Deep Learning model, when the item does not appear in the training data, the item embedding degenerated to the semantic embedding without any collaborative signal [41]. Researchers proposed a DropoutNet to address the cold start problem in recommender systems [40]. By treating cold start as the missing preference data, DropoutNet modified the learning procedure with dropout techniques to explicitly condition the model for missing inputs. Besides the application scenario, we also differ greatly from these works as we reformulate the hybrid recommendation with new items from a graph perspective. Therefore, our proposed model could better capture the relationships between users and items for recommendation, especially for users who have limited records.

5.3 Graph Learning Models and Applications in Recommendation

Recently, Graph Convolutional Networks (GCN) have shown huge success for graph representation learning [20, 38]. These models generate node embeddings in a message passing or information propagation manner in the graph, with a node embedding is recursively computed by aggregating neighboring nodes' information. In fact, researches have already shown that GCNs could be seen as a special kind of Graph Laplacian smoothing [21, 25]. With the requirement of quickly generating embeddings for unseen nodes, GraphSAGE provides an inductive learning approach that learns a mapping function from node features and node links to node embedding [12]. Due to the huge success of the GCNs, several models have attempted to utilize the idea of GCNs for recommendation, such as user-item bipartite graph in collaborative filtering [42, 46], user influence diffusion in social recommendation [45], and item-item similarity graph for similar item recommendation [48]. Most of these GCN based recommendation models are based on CF, and could not tackle the new item problem [42, 45, 46]. Specifically, PinSage is one of the state-of-the-art inductive content based GCN [48]. By taking both item-item correlation graph and item features as input, PinSage learned transformed item embedding with graph convolutional operations. However, in the real-world, new items do not have user ratings, i.e., the node neighbors for embedding learning. Our main technical contribution lies in transferring the knowledge learned from the graph neural networks, such that our

proposed TransGRec framework is also applicable to the new item without any link information.

6 CONCLUSION

In this paper, we designed a TransGRec framework for personalized video highlight recommendation. We based TransGRec on an graph neural network model, and proposed to propagate user embeddings in the graph to alleviate the cold-start user problem. We further proposed a transfer network that learns to transform the item content embedding to the graph neural network space. Therefore, TransGRec is an inductive graph based recommendation approach. Please note that, though we use the personalized video recommendation as an application scenario, our proposed model is generally applicable to any content based recommendation tasks. Finally, extensive experimental results on a real-world dataset clearly showed the effectiveness of our proposed model under various evaluation metrics. In the future, we would like to explore how to design more sophisticated transfer network for graph embedding approximation. Besides, we would like to apply and validate the effectiveness of our proposed framework in the general recommender systems.

ACKNOWLEDGEMENTS

This work was supported in part by the National Natural Science Foundation of China(Grant No.61725203, 61972125, U1936219, 61722204, 61932009 and 61732008).

REFERENCES

[1] Geoffrey E. Hinton Alex Krizhevsky, Ilya Sutskever. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*. 1097–1105.
[2] Grigory Antipov, Moez Baccouche, and Jean-Luc Dugelay. 2017. Face aging with conditional generative adversarial networks. In *ICIP*. 2089–2093.
[3] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. 2017. A generic coordinate descent framework for learning from implicit feedback. In *WWW*. 1341–1350.
[4] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *JMLR* 7, Nov (2006), 2399–2434.
[5] Eric P. Xing Bin Zhao. 2014. Quasi real-time summarization for consumer videos. In *CVPR*. 2513–2520.
[6] Rose Catherine and William Cohen. 2017. Transnets: Learning to transform for recommendation. In *Recsys*. 288–296.
[7] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *NIPS*. 2172–2180.
[8] Rob Fergus Lorenzo Torresani Manohar Paluri Du Tran, Lubomir D. Bourdev. 2015. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*. 4489–4497.
[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*. 2672–2680.
[10] Michael Gygli. 2018. Ridiculously fast shot boundary detection with fully convolutional neural networks. In *CBMI*. 1–4.
[11] Michael Gygli, Yale Song, and Liangliang Cao. 2016. Video2gif: Automatic generation of animated gifs from video. In *CVPR*. 1001–1009.
[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*. 1024–1034.
[13] Ruining He and Julian McAuley. 2016. VBPR: visual bayesian personalized ranking from implicit feedback. In *AAAI*. 144–150.
[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2014. Distilling the knowledge in a neural network. In *NIPS Workshop*.
[15] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2, 5 (1989), 359–366.
[16] Min Hou, Le Wu, Enhong Chen, Zhi Li, Zheng Vincent W., and Qi Liu. 2019. Explainable Fashion Recommendation: A Semantic Attribute Region Guided Approach.. In *IJCAI*. 4681–4688.
[17] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 2012. 3D convolutional neural networks for human action recognition. *TPAMI* 35, 1 (2012), 221–231.
[18] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. 2014. Large-scale video classification with convolutional neural networks. In *CVPR*. 1725–1732.
[19] Aditya Khosla, Raffay Hamid, Chih-Jen Lin, and Neel Sundaresan. 2013. Large-scale video summarization using web-image priors. In *CVPR*. 2698–2705.
[20] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.
[21] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*.
[22] Joonseok Lee and Sami Abu-El-Haija. 2017. Large-scale content-only video recommendation. In *ICCV*. 987–995.
[23] Yong Jae Lee and Kristen Grauman. 2015. Predicting important objects for egocentric video summarization. *IJCV* 114, 1 (2015), 38–55.
[24] Chenyi Lei, Dong Liu, Weiping Li, Zheng-Jun Zha, and Houqiang Li. 2016. Comparative deep learning of hybrid representations for image recommendations. In *CVPR*. 2545–2553.
[25] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*. 3538–3545.
[26] Defu Lian, Qi Liu, and Enhong Chen. 2020. Personalized Ranking with Importance Sampling. In *Proceedings of The Web Conference 2020*. 1093–1103.
[27] Defu Lian, Haoyu Wang, Zheng Liu, Jianxun Lian, Enhong Chen, and Xing Xie. 2020. LightRec: A Memory and Search-Efficient Recommender System. In *Proceedings of The Web Conference 2020*. 695–705.
[28] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. [n. d.]. Unifying distillation and privileged information. In *ICLR*.
[29] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *JMLR* 9, Nov (2008), 2579–2605.
[30] Luc Van Gool Michael Gygli, Helmut Grabner. 2015. Video summarization by learning submodular mixtures of objectives. In *CVPR*. 3090–3098.
[31] Steven M. Seitz Min Sun, Ali Farhadi. 2014. Ranking domain-specific highlights by analyzing edited videos. In *ECCV*. 787–802.
[32] Ana Molino and Michael Gygli. 2018. PHD-GIFs: Personalized Highlight Detection for Automatic GIF Creation. In *MM*. 600–608.
[33] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *TKDE* 22, 10 (2009), 1345–1359.
[34] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.
[35] Pablo Castells RocÃŋo CaÃśamares. 2018. From the PRP to the Low Prior Discovery Recall Principle for Recommender Systems. In *SIGIR*. 1081–1084.
[36] Peijie Sun, Le Wu, and Meng Wang. 2018. Attentive Recurrent Social Recommendation. In *SIGIR*. 185–194.
[37] Peijie Sun, Le Wu, Kun Zhang, Yanjie Fu, Richang Hong, and Meng Wang. 2020. Dual Learning for Explainable Recommendation: Towards Unifying User Preference Prediction and Review Generation. In *WWW*. 837–847.
[38] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph Convolutional Matrix Completion. In *ICLR*.
[39] Dieleman S. Van den Oord, A. and B. Schrauwen. 2013. Deep content-based music recommendation. In *NIPS*. 2643–2651.
[40] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. Dropoutnet: Addressing cold start in recommender systems. In *NIPS*. 4957–4966.
[41] Wang N. Wang, H. and D. Y. Yeung. 2015. Collaborative deep learning for recommender systems. In *KDD*. 1235–1244.
[42] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *SIGIR*. 165–174.
[43] Le Wu, Lei Chen, Richang Hong, Yanjie Fu, Xing Xie, and Meng Wang. 2019. A hierarchical attention model for social contextual image recommendation. *TKDE* (2019).
[44] Le Wu, Lei Chen, Yonghui Yang, Richang Hong, Yong Ge, Xing Xie, and Meng Wang. 2019. Personalized Multimedia Item and Key Frame Recommendation. In *IJCAI*. 1431–1437.
[45] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A Neural Influence Diffusion Model for Social Recommendation. In *SIGIR*. 235–244.
[46] Yuexin Wu, Hanxiao Liu, and Yiming Yang. 2018. Graph Convolutional Matrix Completion for Bipartite Edge Prediction. In *KDIR*. 51–60.
[47] Ting Yao, Tao Mei, and Yong Rui. 2016. Highlight detection with pairwise deep ranking for first-person video summarization. In *CVPR*. 982–990.
[48] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *KDD*. 974–983.
[49] Chao W. L. Sha F. Zhang, K. and K. Grauman. 2016. Video summarization with long short-term memory. In *ECCV*. 766–782.