SSR: Joint Optimization of Recommendation and Adaptive Bitrate Streaming for Short-form Video Feed

Dezhi Ran†, Yuanxing Zhang†, Wenhan Zhang†, Kaigui Bian†‡
†*School of Electronics Engineering and Computer Science, Peking University, Beijing, China*
‡*National Engineering Laboratory for Big Data Analysis and Applications, Beijing, China*
Email: {dezhiran, longo, pku_zwh, bkg}@pku.edu.cn

***Abstract*—Short-form video feed has become one of the most popular ways for billions of users to interact with content, where users watch short-form videos of a few seconds one-by-one in a session. The common solution to improve the quality of experience (QoE) for short-form video feed is to treat it as a common sequential item recommendation problem and maximize its click-through rate prediction. However, the QoE of short-form video streaming under dynamic network conditions is jointly determined by both recommendation accuracy and streaming efficiency, and thus merely considering recommendation will lead to the degradation of the QoE of the streaming system for the audience. In this paper, we propose SSR, namely the short-form video streaming and recommendation system, which consists of a Transformer-based recommendation module and a reinforcement learning (RL) based bitrate adaptation streaming module. Specifically, we use Transformer to encode the session into a representation vector and recommend proper short-form videos based on the user's recent interest and the timeliness characteristics of short-form video contents. Then, the RL module combines the representation of session and other observations within the playback, and yields the appropriate bitrate allocation for the next short-form video to optimize a given QoE objective. Trace-driven emulations verify the efficiency of SSR compared to several state-of-the-art recommender systems and streaming strategies with at least 10%-15% QoE improvement under various QoE objectives.**

***Keywords*-Short-form video feed, adaptive video streaming, quality of experience, recommendation**

I. Introduction

By 2022, 82% of all mobile data will be consumed by videos [1]. Short-form video feed has become one of the most popular ways for mobile users (especially for the young generation) to interact with content. Many video service providers have launched short-form video services such as TikTok, Vine, and Likee, which allow billions of users to generate and share their own stories via videos of a few seconds. Many mobile applications present lists of content in waterfall layout, where users should select and click the ones they are willing to watch. In contrary to these applications, the short-form video feed displays the short-form videos one-by-one in full-screen, and thus users would just decide whether to watch the current short-form video displayed to them. The user can choose to watch the whole video if she/he likes the video, or skip it at any time, after which the next video is auto-played to her/him.

The proliferation of short-form video feed has attracted extensive research effort, mainly on transforming the short-form video delivery service into the sequential recommendation problem [2]. Accurate identification of user interests can improve the extent that users are engaged in the application. In practice, to relieve the server-side overload on responding massive concurrent requests from users, video service providers would provide list-wise recommendation [3] to recommend the client-side playback to follow a specific list of short-form videos, and thus the heavy computation models on the server could be invoked less [4]. However, streaming is another evident factor beyond the list-wise recommendation that could affect the quality of experience (QoE) on video playback [5], such as the dynamic network conditions that may lead to rebuffering. What is worse, adaptive streaming mechanisms for long-form videos may not work well under the unique settings of the short-form video feed, nor do they cooperate with the list-wise recommender systems. As the simplest solution, industries have deployed a mechanism of replaying and streaming constant low-bitrate short-form videos at the expense of lowering the QoE (e.g., video resolution). Specifically, the client-side QoE would inevitably be degraded as such a solution fails to consider the performance of streaming under dynamic network conditions, or cooperate with the list-wise recommender system.

In summary, previous short-form video recommending strategies fail to address the following challenges for short-form video feed.

- **Dynamic network conditions**. Client-side network settings and server-side overload (especially during peak hours) determine the network condition when users watch short-form videos. Streaming high definition short-form videos under poor network conditions leads to rebuffering of the playback, which would degrade the QoE of users.
- **Timeliness for short-form video contents**. To capture audiences' attention, user-generated short-form video contents usually has strong timeliness, e.g., related to current social events, fashion. Unlike long-form videos,

it is not attractive to recommend short-form videos released a long time ago to users.

- **Right timepoint to recommend next list of videos**. During the playback of current list of videos, it is important to choose a right timepoint to start recommending the next list. For instance, a late recommendation after the playback of current list may lead to an empty buffer, incurring rebuffering when waiting for playing the next list of videos; an early recommendation during the playback of current list may lose opportunities of observing explicit users' feedback of the playback of current list, leading to lower recommendation accuracy of next list of videos.

To address the above challenges, we explicitly formulate the joint optimization of recommendation and streaming (JORS) problem for short-form video feed, and design the solution to the problem for achieving a high QoE of the playback. Our preliminary findings reach out that the released time of the video is a significant piece of evidence to estimate whether users would like the video, and recommendation over explicit feedback towards videos requires high-order feature investigation for accuracy concern. Besides, enlightened by the promising performance of reinforcement learning (RL) models on the general video streaming task [6], RL can be used to integrate the recommendation and streaming strategies, while coping with the dynamic network conditions and various QoE objectives.

In this paper, we propose SSR (Short-form video Streaming and Recommendation), a short-form video streaming system which combines recommendation with adaptive bitrate streaming techniques. SSR applies Transformer [7] to model the user preferences based on the previous explicit feedback to make recommendation and preference encoding, where contextual user behaviors, short-form video attributes, the released time of the videos are embedded as the input to the Transformer module. SSR then employs RL strategy to dynamically decide when to perform the recommendation and which bitrate should be allocated to each short-form video to optimize the given QoE objectives. We construct a dataset over real-world short-form video interactions and the public bandwidth traces to examine the performance of SSR. Compared to several state-of-the-art recommender systems and streaming systems, SSR could improve the recommendation accuracy by at least 5%, and improve the users' perceived QoE by 10%-15%. Extensive analysis reveals that SSR accurately identifies the user interest and makes effective buffer control during playback. Experimental results indicate that SSR is practical to be implemented in the real-world short-form video feed applications.

II. Problem Formulation

In this section, we formulate the problem of joint optimization of recommendation and streaming (JORS) for short-form video feed, and introduce the QoE objectives to be optimized. Fig. 1 illustrates a short-form video feed application, which decides the bitrate for the next video and the time to request recommendation, and receives the list-wise recommendation from the server.

A. Playback of Short-form Video Feed

Short-form video feed playback task. Let U denote the set of users, with $|U|$ users in total. Suppose that the server prepares a set of short-form videos, V, to be recommended to users, with $|V|$ short-form videos. Denote the length of video $v \in V$ as l_v, which falls in 10 seconds to 30 seconds for short-form videos. Different from the long-form video streaming that divides the videos into chunks (short segments of consecutive video frames) or tiles (spatial fragments in a chunk), short-form video service providers encode the entire short videos by a set of candidate bitrates $\mathcal{BR}$ to serve for various server-side and client-side network conditions. For each user $u \in U$, denote $\mathbf{v}_u$ her *watch list*, according to which the player downloads short-form videos in the order to display. After the playback of each $v \in \mathbf{v}_u$, user u would explicitly expose attitudes (or preferences) towards the short-form video, i.e., if the user likes a video, she would probably fully-watch it (or click "like" button); otherwise, she tends to swipe up (i.e., dislikes) the video and begins watching the next video. Implicitly, we denote the attitude $I_{u,v}$ of video v by user u as an indicator, where "1" stands for "like", and "0" stands for "dislike". Meanwhile, we could use the watching duration of a video as the explicit feedback from users. Let $w_{u,v}$ denote the watching duration on video v of user u. Following the statistical analysis in [8], we assume that the watching duration $w_{*,v}$ of short-form video v follows the truncated exponential distribution. Therefore, in this paper, we formulate the explicit feedback $p_{u,v}$ from user u towards video v into logarithmic representation as:

$$p_{u,v} = \log_{\lambda l_v}(\max\{w_{u,v} - \alpha, \epsilon\}), \tag{1}$$

where λ and α are empirical parameters from the datasets. Here, ϵ is a threshold indicating that the user will immediately skip the video, and we fix $\epsilon = \frac{1}{\lambda l_v}$ in this paper. The short-form video feed decision model should be insensitive to these parameters.

Considering the heavy load on the server by massive requests from clients and the potential rebuffering due to the delay after the drain of the watch list, a common solution is to recommend by list [9], instead of reacting instantly to deliver one single video for next playback. Denote $Y_{u,j} = \{y_{u,j,1}, y_{u,j,2}, \ldots, y_{u,j,L}\}$ the j-th video *recommended list* to user u of (fixed) length L. Collecting the explicit feedback ("like" or "dislike") after the playback of the entire video list provides most evidence for generating the next recommended list, while requesting recommendation ahead of the end of the list significantly reduces the risk of rebuffering. To avoid termination of the on-going downloading thread

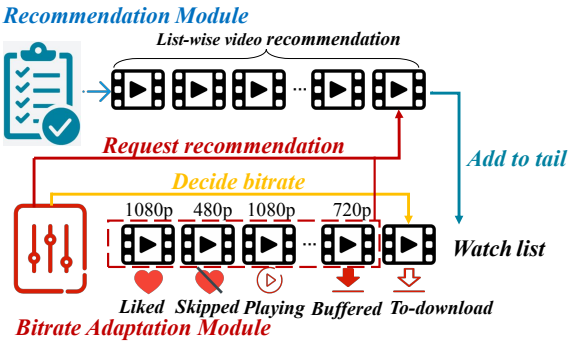


Figure 1: Collaboration between the bitrate adaptation streaming and the recommendation modules.

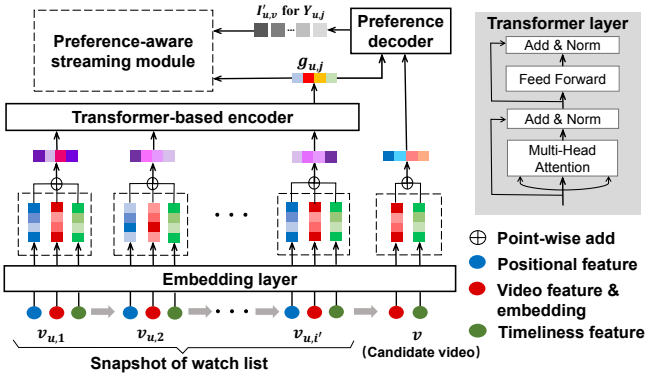


Figure 2: Transformer-based recommendation and preference prediction modules.

and take full advantage of the bandwidth, the request for the recommended list would be sent along with the start of downloading a video in the current watch list. In this case, it is essential to find the proper time to obtain the list. Then, the newly-received recommended list would be appended to the watch list, while the remaining part of the previous recommended list can be seen as an exploration to capture the user preference.

Streaming process of the short-form video feed. Suppose the client begins to download the i-th video $v_{u,i}$ from the watch list $\mathbf{v}_u$ with bitrate $br_{v_{u,i}} \in \mathcal{BR}$ at timestamp $t_{u,i}$. Considering the video CODEC, denote the actual size of the video as $q_{u,i}(br_{v_{u,i}})$, which requires a duration of $t^d_{u,i}$ to be downloaded under the dynamic network conditions. In particular, requesting the recommended list requires a short delay for the response, denoted as t^S, which can be executed in parallel with the downloading process.

Assume the size of the playback buffer is $B_{\max}$, measured in mega-Bytes. The total size of the buffered video content must not exceed this limitation. Define $B_u(t)$ as the remaining playback time of the video content in the buffer for user u, i.e., *buffer occupancy*, at timestamp t. The buffer occupancy decreases along with video playback, and increases by $l_{v_{u,i}}$ when the i-th short-form video is downloaded completely. Let $B_{u,i} = B(t_{u,i})$ be the buffer occupancy when the request for $v_{u,i}$ is sent. Note that the client drops the remaining playback of the video in the buffer when the users skip the current video. Therefore, the buffer occupancy at $t_{u,i+1}$ could be calculated by:

$$B_{u,i+1} = \rho\Big(\rho\Big(B_{u,i} - t^d_{u,i} - t^s_{u,i}\Big) + l_{v_{u,i}} - \mathbb{I}[i = |\mathbf{v}_u|]t^S\Big) \tag{2}$$

where $t^s_{u,i}$ represents the total length of the elapsed video length between $t_{u,i}$ and $t_{u,i+1}$ (including both watching duration and the skipped parts of the videos), and ρ(x) = max(x, 0) which guarantees that buffer occupancy is no less than 0.

Session-based short-form video recommendation. The watch list of a user forms a session, which can be divided into three parts: the videos already been displayed, the videos cached in the buffer, and the videos to be downloaded. Along with the playback of the short-form videos, users expose explicit feedback to the displayed videos. Let $\mathbf{vp}_{u,i} = \{(v_{u,1}, p_{u,v_{u,1}}), (v_{u,1}, p_{u,v_{u,2}}), \ldots\}$ denote the snapshot of the displayed videos with feedback from user u's watch list $\mathbf{v}_u$ at timestamp $t_{u,i}$ (note that the request for recommendation can only be sent at the time when the clients begin to download a short-form video). Suppose that the client has received $(j-1)$ recommended list in the previous playback procedure, i.e., $|\mathbf{v}_u| = (j-1)L$. The recommender system should yield the next recommended list $Y_{u,j}$ based on $\mathbf{vp}_{u,i}$. The recommended list will be extended to the watch list, and then the client would follow the list to download and display the corresponding short-form videos. The recommendation should yield evidence for the balance between accurate preference prediction and efficient streaming.

Bitrate allocation for short-form video feed. Playback of short-form videos is usually under dynamic network conditions, where the bottleneck may be on either server-side or client-side. The streaming module of the application would attempt to allocate a bitrate for the next short-form video by the perceived bandwidth to maximize the QoE of users, i.e., assigning $br_{v_{u,i}} \in \mathcal{BR}$ to the i-th video from the watch list based on the previous playback statistics. Intuitively, when the bandwidth is high and stable, the streaming module tends to request as many short-form videos with high bitrate as possible. As the buffer size is limited in the short-form video feed application, downloading videos with higher bitrate means that fewer videos can be buffered, leading to higher risk of rebuffering. Meanwhile, when the network condition is poor, the streaming module would coordinate with the recommender system and request videos with low bitrate and even in short-length (i.e., short period to download) or high compression rate (e.g., animation) to avoid the potential rebuffering (or replaying in the industrial solution).

B. Quality of Experience Metrics

The quality of experience (QoE) is a measurement of the attitude of users towards the applications, which is precisely the optimization objective of the short-form video feed streaming system. Conventionally, there are quality-of-service (QoS) metrics indicating for QoE of the playback of long-form videos, as well as accuracy metrics showing the quality of recommendation. In the short-form video feed applications, these two kinds of metrics both significantly affect the QoE of the playbacks, which should be considered together to evidence for QoE of users. Specifically, we define the following metrics for the QoE of short-form video feed streaming:

1) *Effective bitrate*. The downloaded short-form videos should be in high resolution, or users may view the low-bitrate videos. Meanwhile, the displayed content should be consistent with the user preference, otherwise the bandwidth is wasted owing to the skipping of the users. Therefore, we expect to take full advantage of the bandwidth for effective bitrate:

$$\mathrm{QoE}^{a}_{u,i} = I_{u,v_{u,i}} \log q_{u,i}(br_{v_{u,i}}). \tag{3}$$

2) *Recommendation efficiency*. The recommender system should present the videos with the most positive attitudes in the recommended list to maximize the explicit feedback in the future playback:

$$\mathrm{QoE}^{t}_{u,i} = p_{u,v_{u,i}}. \tag{4}$$

3) *Rebuffering*. Since rebuffering heavily degrades the user experience, we take the rebuffering time as a penalty factor:

$$\mathrm{QoE}^{r}_{u,i} = \rho(t^{d}_{u,i} + t^{s}_{u,i} - B_{u,i}). \tag{5}$$

Then, the QoE objective of $v_{u,i}$ from the watch list could be formulated by:

$$\mathrm{QoE}_{u,i} = \mu_1 \mathrm{QoE}^{a}_{u,i} + \mu_2 \mathrm{QoE}^{t}_{u,i} - \mu_3 \mathrm{QoE}^{r}_{u,i}, \tag{6}$$

where $\boldsymbol{\mu} = (\mu_1, \mu_2, \mu_3)$ are non-negative weighting parameters. Note that different settings of $\boldsymbol{\mu}$ correspond to various preferences of users and result in different optimization targets. For example, if a user is sensitive to the video content and skips the videos quickly against her interest, μ_2 should be relatively raised.

To provide high QoE for the solution to the JORS problem, we propose a rebuffering-aware recommendation mechanism that yields the recommendation in real time and maximizes the overall QoE objective over the entire watch list.

III. Design of SSR

SSR model consists of two major modules to cooperatively solve the JORS problem. The Preference Prediction Module predicts the user preference over the watch list and generates the recommendation video lists, while the Bitrate Adaptation Module performs dynamic bitrate allocation with reinforcement learning based on the monitored playback statistics. The objective of SSR is to maximize users' QoE over their watch lists.

A. Preference Prediction Module

Given the snapshot $\mathbf{vp}_{u,i'}$ of the watch list $\mathbf{v}_u$ when requesting for the j-th recommended list $Y_{u,j}$. We employ a Transformer-based recommender system, which tackles the contextual user behaviors, user portrait, and video attributes to select the top-L videos at the timestamp for the user. The network architecture of preference prediction module is shown in Fig. 2.

Transformer-based sequence encoder. In SSR, we utilize the *embedding layer* to encoder each video v into a d_e-dimension embedding, denoted as e_v. Meanwhile, as the cover image of a short-form video usually spotlights its main content to the audience, we retrieve a 64-d vector ι_v from video v as the video-relevant feature. Besides, inspired by [10], we use feature enhancement over user u's attitude towards each video $v_{u,i}$ by $\tau_{v_{u,i}} = [w_{u,v_{u,i}}, l_{v_{u,i}}, I_{u,v_{u,i}}, \sqrt{w_{u,v_{u,i}}}, w^2_{u,v_{u,i}}, \sqrt{l_{u,v_{u,i}}}, l^2_{u,v_{u,i}}, 1 - I_{u,v_{u,i}}]$. Then, we obtain the representation table $\mathbf{E}$ of $d = d_e + 72$ dimension for all videos, and the representation of video v is $\mathbf{E}(v) = [e_v; \iota_v; \tau_v]$, where "[;]" represents concatenation. In SSR, we apply Transformer to encode the session, and thus we follow the setting in [7], [11] to involve positional embedding pos_i of dimension d for $1 \le i \le |\mathbf{vp}_{u,i'}|$ to maintain the sequential influence. Considering the timeliness characteristics of short-form videos, we encode the released time of each video v into a static representation ζ_v of dimension d initialized by sine and cosine in minute-granularity over a long time period. Given a snapshot $\mathbf{vp}_{u,i'}$, we could find the last video $v^*_{\mathbf{vp}_{u,i'}}$ that the user likes (if not exist, label by a special video id). We assume that the user would probably prefer the videos that were published near the released time of $v^*_{\mathbf{vp}_{u,i'}}$. In this case, we set the inputs to the Transformer as $\mathbf{X}_{u,1} \triangleq \{\mathbf{E}(v_{u,i}) + pos_i + \zeta_{v^*_{\mathbf{vp}_{u,i'}}}\}$ as the description of the session with user behavior and the short-term interest. Suppose that we use Ψ layers of Transformer to encode the sequence. The Transformer layers apply the scaled-dot product (SDP) [7] to measure the correlation among videos in the sequence, regarding the ψ-th layer $\mathrm{SDP}_{\psi,h}(\mathbf{X}_{u,\psi})$, given $\mathbf{X}$ which is the input embedding or the output of the previous layer. We then deploy the multi-head attention, i.e., H heads in total, to capture different high-order feature maps of user characteristics:

$$\mathbf{Z}_{u,\psi} = MH(\mathbf{X}_{u,\psi}) = [\mathrm{head}_{\psi,1}; \mathrm{head}_{\psi,1}; ...; \mathrm{head}_{\psi,H}]\mathbf{W}^{\mathbf{H}}_{\psi}, \tag{7}$$

and the h-th head is computed by:

$$\mathrm{head}_{\psi} = \mathrm{SDP}_{\psi,h}(\mathbf{X}_{u,\psi}),\ h \in \{1, 2, \ldots, H\}, \tag{8}$$

where $\mathbf{W}_{\psi}^{\mathbf{H}} \in \mathbb{R}^{d \times d}$ is the learnable parameter. To enhance model with non-linearity, Transformer uses *point-wise feed-forward network (FFN)* with layer normalization. We also add residual connections to further improve the performance of the Transformer layer, as shown in Fig. 2. We regard the output vector at the last position of the last layer $\mathbf{F}_{u,\Psi}$, i.e., denoted as $g_{u,j} \triangleq F_{u,\Psi,|\mathbf{vp}_{u,i'}|}$, as the user preference vector representing the user's preference and interest towards the previous watch list, which will be forwarded to RL as part of the observation.

User preference decoder. We use a Multi-Layer Perception (MLP) with sigmoid activation ($\sigma(\cdot)$) to decode the sequence and predict the implicit feedback:

$$I'_{u,v} = \sigma(w_I^T(W_g g_{u,j} + W_c(\mathbf{E}(v) + \zeta_v))). \tag{9}$$

where $w_I \in \mathbb{R}^{d \times 1}$, $W_g, W_c \in \mathbb{R}^{d \times d}$ are the parameters of the decoder. The preference prediction model is trained individually on cross entropy loss between $I_{u,v_{u,i}}$ and $I'_{u,v_{u,i}}$ over the entire snapshot of the watch list for $u \in U$. In practice, the server-side preference prediction module would send the final outputs of the Transformer blocks and the predicted implicit feedback of the short-form videos in the recommended list back to the client.

B. Preference-aware Streaming Module

The client-side preference-aware short-form video streaming module allocates bitrate to each short-form video following the watch list based on the previous playback statistics. When the module selects the bitrate for the next short-form video, it should also decide whether to request for the recommended list simultaneously. An early or late request for recommendation leads to either the recommendation deviated from users' true interest or the risk of rebuffering. As users may present various preferences on the QoE metrics, the streaming module should cooperatively determine the bitrate for the videos in the watch list and the time to request recommendation. To improve the overall QoE of users, we utilize reinforcement learning to conduct and optimize dynamic streaming control.

Streaming with reinforcement learning. In the standard RL settings, an agent interacts with an environment $\mathcal{E}$ over a number of discrete time steps. At each timestamp $t_{u,i}$, the agent receives an observation $o_{u,i}$, transforms the observation tot state $s_{u,i}$, and takes an action $a_{u,i}$ according to its policy π, and receives a scalar reward $r_{u,i}$. The discounted return $R_{u,i} = \sum_{\kappa=i}^{|\mathbf{v}_u|} \gamma^{\kappa-i} r_{u,\kappa}$ is the total accumulated reward from time step $t_{u,i}$ with discount factor $\gamma \in (0, 1]$. The goal of reinforcement learning is to maximize the expected return by optimizing the policy.

Apparently, the streaming control can be integrated into the RL framework. Specifically, the client of the playback is seen as the agent, and the streaming system would be regarded as the environment. When the client finishes downloading the i-th video, it collects the playback statistics as the observation $o_{u,i}$. The client updates the previous state $s_{u,i-1}$ with $o_{u,i}$ and gets the current state $s_{u,i}$. Then, it makes a decision (action) a_i from discrete action set $\mathbf{A} = \{0, 1\} \times \mathcal{BR}$ (indicating whether to request recommendation and which bitrate to select) based on the learnable streaming control policy $\pi_\theta(a_{u,i}|s_{u,i})$ (parameterized by θ), and receives the user QoE as the reward $r_{u,i} = QoE_{u,i}$. The goal of the streaming control here is to learn the optimal policy π_θ^* that maximizes the expectation of the total discounted QoE:

$$\pi_{\boldsymbol{\theta}}^* = \arg\max_{\pi_{\boldsymbol{\theta}}} \mathbb{E}_{a_{u,*} \sim \pi_{\boldsymbol{\theta}}} Q^{\pi_{\boldsymbol{\theta}}}(s_{u,1}, a_{u,1}), \tag{10}$$

where $Q^{\pi_{\boldsymbol{\theta}}}(s_{u,i}, a_{u,i}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}[R_{u,i}]$ is the *action-value function*, i.e., the expected return beginning from state $s_{u,i}$ and action $a_{u,i}$, under policy $\pi_{\boldsymbol{\theta}}$. Note that the Eqn. (10) is a transformation of the objective. Hence, we could embed the streaming control into the framework of reinforcement learning. Finally, after training the model with RL algorithms, e.g., Advantage Actor-Critic (A2C) [12] in SSR, the client only needs to pick the action with the highest probability during playback.

Bitrate allocation model. In SSR, the client records the bandwidth of past 10 seconds before $t_{u,i}$, i.e., $\mathbf{N}_{u,i} = (N(t_{u,i}-9), N(t_{u,i}-8), ..., N(t_{u,i}))$, to capture the dynamic information, where $N(t_{u,i})$ denotes the bandwidth at timestamp $t_{u,i}$. the client also collects $B_{u,i}$, $I_{u,v_{u,i-1}}$ (or $I'_{u,v_{u,i-1}}$ if not yet played), $l_{v_{u,i}}$, $q_{u,i}(\mathcal{BR})$, $\iota_{v_{u,i}}$, and $e_{v_{u,i}}$ to describe the current playback status with the next video attribute. For the consideration of playback fluctuation, the client should be aware of the previous action, i.e., $a_{u,i-1}$. Besides, to give preference-aware control, the output of the Transformer encoder on the last requested recommended list $g_{u,j}$ are also added to the observation. The observation helps re-evaluate the temporal user interest and depict the current playback condition. To encode the entire historical observations, we utilize a gated recurrent unit (GRU) network, a variant of RNN for reducing the vanishing gradient problem [13]. The state s_i can then be calculated by $s_{u,i} = \text{GRU}(s_{u,i-1}, o_{u,i})$. Note that the observations are the consequences of previous decisions, so the actions are implicitly included in the state.

We conduct the streaming control based on the policy $\pi_\theta(a_{u,i}|s_{u,i})$. In concrete, π_θ utilizes a linear project transformation and a softmax layer to convert state $s_{u,i}$ to the discrete action distribution. As SSR uses the A2C [12] algorithm to optimize the model, an extra Q-network (realized by a fully-connected layer) is applied on the state to generate the estimation of the state-action-value function, namely $Q(s_{u,i}, a_{u,i}; W^C)$ with parameter W^C.

A2C-based model training. SSR exploits A2C algorithm to train the streaming control module for optimizing the QoE objectives. The A2C algorithm is entirely online and incremental, which can be updated during playback.

Specifically, at each time $t_{u,i}$ to make decision, the client constructs the sequence of $o_{u,i}, s_{u,i}, a_{u,i}, r_{u,i}, o_{u,i+1}, s_{u,i+1},$

and computes the Q-target:

$$\tau_{u,i} = r_{u,i} + \gamma \max_{a'} Q(s_{u,i+1}, a'). \tag{11}$$

Then, the gradient of W^C is calculated by minimizing the L2 loss between $\tau_{u,i}$ and $Q(s_{u,i}, a_{u,i}; W^C)$:

$$dW^C = \nabla_w (\tau_{u,i} - Q(s_{u,i}, a_{u,i}; W^C))^2. \tag{12}$$

Note that both the Q-target $\tau_{u,i}$ and $Q(s_{u,i}, a_{u,i}; W^C)$ are the estimations of state-action-value function, where Q-target is a more precise estimation due to the one-step look ahead. Thus, the Q-network will be more close to the actual state-action-value function after parameter update.

After that, the client calculates the Q-error:

$$\delta_{u,i} = r_{u,i} + \gamma \max_{a'} Q(s_{u,i+1}, a'; W^C) - Q(s_{u,i}, a_{u,i}; W^C), \tag{13}$$

which is an estimation of the advantage. Based on $\delta_{u,i}$, the gradient of θ can be deduced by:

$$d\theta = \nabla_\theta \pi_\theta(s_{u,i}, a_{u,i}) \delta_{u,i}. \tag{14}$$

It is worth mentioning that the Q-error is equal to the gradient of the L2 loss between $\tau_{u,i}$ and $Q(s_{u,i}, a_{u,i}; W^C)$ on Q function, while the update of θ is equal to the gradient ascent on the expected value of $Q^{\pi_\theta}(s_{u,i}, a_{u,i})$.

Discussion of the interaction between two modules. As the recommender module runs on server and the streaming module runs on clients, the two stages of the framework cooperate with each other. By $g_{u,j}$, which determines the bitrate of the videos in both the remaining part of the previous rec list and the next rec list. By RL module and GRU model, which determines the time to request recommendation and rectify the learnt preference of videos in the rec list (similar to running a GRU4REC model locally). The two components are trained together offline, and run online over server and clients. In this case, the recommender module will embed as much information as possible in $g_{u,j}$, and the streaming module can yield corresponding actions. The recommender module is pre-trained to accelerate training in practice.

IV. Evaluation

A. Settings

Dataset construction. We select watch lists and bandwidth traces from public datasets and combine them for the evaluation.

Videos. We choose 3,000,000 videos with 10,000 users from the real word short-form video dataset [1], 90% of which are of 10-20 second length. We assume that there are three candidate bitrates for short-form video in total, where each video is encoded by 1Mbps (480p), 3Mbps (720p), and 10Mbps (1080p). For each user, we construct candidate list of length at least 48 from the user playback history. In each list, a number of 13-20 videos are "liked" or fully-watched by the user. 10% of the watch lists are selected as the test set, which are guaranteed not to appear in the training procedure. We conduct preliminary measurements on the dataset and find that the distribution of the watch time of each selected video follows the exponential distribution.

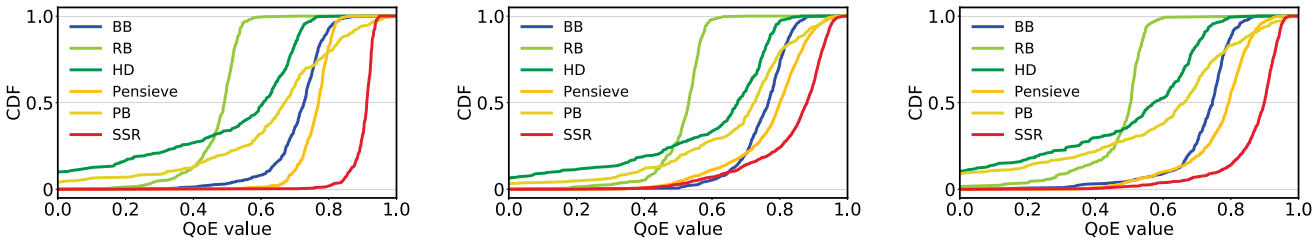
Bandwidth traces. We choose 428 bandwidth traces with various bandwidth patterns from public datasets [14], [15] to emulate different network bandwidth conditions. The bandwidth ranges from 128kbps to 12Mbps. Similar to watch lists, we randomly select 80 bandwidth traces as the test set for evaluation.

Implementation details of SSR. We set (λ, α) to (0.6, 2) to transform watching time to preference, which best fits the distribution of watching time in the dataset. We set the length of the video list L to 8, i.e., each list consists of 8 short-form videos. In the preference prediction module, the size of input embedding d is set to 96, i.e., $d_e = 24$. We find that a higher dimension of embedding would not further improve the performance. We test the performance of different numbers of self-attention layers of SSR and fix the layer number Ψ to 6 with $H = 8$ heads. We pretrain the Transformer following [16]. Regarding the streaming module, we set the hidden size of GRU as 64 in the bitrate allocation module. We utilize the Adam optimizer [17] to train both modules, where the linear decay on learning rate and gradient clipping are also applied to prevent overfitting. The buffer capacity $B_{\max}$ is set to 40MB of playback time, which is appropriate in real-world applications.

Baseline methods. We propose five streaming baseline models for comparison, cooperating with the preference prediction module in this paper. The baseline models request recommendation along with the downloading of the last second video from the watch list (the best time for the fixed recommendation request). *Rebuffer-based strategy* (RB): RB tries to avoid any chance of rebuffering and will conservatively allocate the lowest bitrate for each short-form video. *High-bitrate strategy* (HD): HD allocates the highest bitrate for each short-form video, ignoring the potential rebuffering. *Bandwidth-based strategy* (BB): BB merely allocates the highest bitrate based on the previous bandwidth condition. *Preference-based strategy* (PB): PB allocates a higher bitrate to the videos when the predicted preference is larger than 0.5, while the other videos are allocated with a low bitrate. *Pensieve*: Pensieve [5] is originally designed to make bitrate adaptation for streaming long-form videos by the RL model, without considering preference on different short-form videos. We modify it to treat each short-form video as a video chunk, and follow conventional DASH streaming systems.

Emulation platform. We conduct trace-driven emulation of the downloading and playing of the short-form videos between server and client. For each user in the dataset, the short-form videos she actually interacts with are taken as the candidate set to her, from which 32 videos (4 lists) will

[1] https://www.kuaishou.com/activity/uimc/datadesc



(a) (5, 8, 0.5) (b) (8, 5, 0.5) (c) (5, 5, 1)

Figure 3: The CDFs of the average QoE values under three QoE objectives generated by the compared algorithms.

be displayed to her. The first recommended list will follow the actual first eight videos according to the dataset. After the playback begins, the application follows the preference prediction module and streaming module, where we record the QoE metrics throughout the playback.

Performance metrics. We select three sets of parameters of $\boldsymbol{\mu}$ to represent three QOE objectives with various preferences. Specifically, the QoE weights are $\boldsymbol{\mu}_1 = (8, 5, 0.5)$, $\boldsymbol{\mu}_2 = (5, 8, 0.5)$, $\boldsymbol{\mu}_3 = (5, 5, 1)$, indicating the preference of maximizing the effective bitrate, maximizing total effective watch time over playback, and minimizing rebuffering time respectively. The QoE objectives of the compared streaming systems are normalized to $[0, 1]$ by the maximal value of the QoE objective, which is calculated under ideal bandwidth (always download the 1080p without rebuffering).

B. Performance of SSR

Performance on JORS problem. We compare the proposed SSR model with the baseline and other state-of-the-art algorithms on all QoE objectives in the settings. Fig. 3 shows the detailed CDF results under three QoE objectives for all the algorithms. In particular, compared to the state-of-the-art streaming systems originally designed for long-form videos, SSR improves the QoE by 10%-15% when seeking effective bitrate in the QoE objective, i.e., $\boldsymbol{\mu}_1$. This improvement implies that SSR takes advantage of the limited bandwidth to allocate high bitrates to the most preferred videos and lower bitrates to the less-preferred ones following the given watch list. When considering most on maximizing the effective watching duration in the QoE objective, i.e.,$\boldsymbol{\mu}_2$, SSR harmonizes the priorities among the three metrics. In addition, SSR suffers from less rebuffering time than the compared approaches when focusing more on the rebuffering time in QoE objective, i.e., $\boldsymbol{\mu}_3$.

Since it is evident that various QoE objectives require inherently different streaming strategies, the algorithms that employ fixed control strategies (e.g., HD, RB, BB, and PB) struggle to optimize different QoE objectives. In contrast, the RL-based methods, e.g., the proposed SSR, could adapt to any QoE objective, and thus outperform the other methods. Meanwhile, although Pensieve and PB algorithms have the capability to adapt to the various QoE objectives, they do not consider the preference of the users such that they can not utilize the bandwidth resources properly compared to the proposed SSR. The above analysis over the results reveals the consistency with our initial design of SSR, coping with the dynamic network conditions and finding the best time to recommend for optimizing various QoE objectives.

Table I: The accuracy of preference prediction module.

Method	POP	STAMP	GRU4REC+	NARM	BST	SSR
AUC	-	0.721	0.748	0.743	0.762	**0.774**
NDCG@8	0.523	0.725	0.763	0.754	0.776	**0.787**

Precision of preference prediction. We examine the performance of the user preference prediction and recommendation model for user preference, compared to the following state-of-the-art recommender systems with $\mathbf{E}(v)$ as inputs: *STAMP* [18] is a session-based recommender using attention mechanism over the session. *GRU4REC+* [19] uses GRU to learn the representation of the session with decent loss function and sampling. *NARM* [20] leverages two separate GRUs to caputure the long / short-term interest and encodes the session by attention. *BST* [11] is a Transformer-based recommender without the unique characteristic of the short-form video. Besides, we also implement the popularity based method (POP) which ranks the videos by the popularity on the entire dataset, to demonstrate the complexity of the dataset. These methods are implemented in the same settings as in this paper, and thus the total number of parameters of each algorithm is similar. As we know the actual feedback of the user to all videos in her candidate set, we conduct two experiments to evaluate the performance of the list-wise recommendation We first record the Area Under the Curve (AUC) by computing the score (ranging from 0 to 1 after sigmoid activation) of the videos in the next recommended list from the dataset over the ground-truth of whether the user likes the videos. Meanwhile, we also collect the Normalized Discounted Cumulative Gain (NDCG) of each recommended list along with the trace-driven emulation via replacing the preference prediction module by the compared recommender

Table II: The QoE of different recommendation schedules.

Method	SSR	1-Rec	2-Rec	3-Rec	4-Rec	5-Rec	6-Rec	7-Rec
Avg. QoE	**0.902**	0.780	0.775	0.804	0.786	0.812	0.766	0.783
NDCG@8	0.787	0.765	0.762	0.779	0.781	0.787	0.787	**0.792**

systems. The recommended list does not intersect with each other, and thereby the NDCG metric would be calculated between the recommended list and the remaining videos in the candidate set.

As shown in Table I, our proposed preference prediction model outperforms the state-of-the-art algorithms in short-form video recommendation. Compared to BST, the remarkable improvement reveals that the timeliness of the short-form videos does contribute the accuracy. The results of precision and ranking accuracy indicate that the proposed algorithm is qualified to be extended to recommend from a large candidate set in practice.

Importance of dynamic recommendation scheduling. We check the function of dynamic recommendation scheduling by comparing it to K-Rec (K = 1,2,...,7), which will heuristically generate the next short-form video list when the user finishes the K-th short-form video in the current list. The QoE weight is fixed to be $\boldsymbol{\mu}_1$ as an example. Shown in Table II, the dynamic recommendation scheduling controlled by RL performs best out of all strategies. The 5-Rec performs better than other baselines, balancing between utilizing the latest user interactions and avoiding rebuffering. Hence, the performance of the baseline streaming models with 5-Rec shown in Fig. 3 can be seen as the upper bound of them.

V. Conclusion

In this paper, we formulate the joint optimization of recommendation and streaming problem for short-form video feed. We decently define the QoE metrics to best describe the user behavior and attitude towards the list-wise recommended short-form videos, which are inherently different from those of long-form videos. To maximize the QoE objectives of the JORS problem, we present SSR, which integrates the preference prediction to the streaming system to make full use of the playback history. SSR employs Transformer blocks over the timeliness characteristics of the short video contents to enhance the accuracy of the recommendation module, which provides evidence for the streaming module to decide when to request the recommendation and which bitrate should be allocated to the next video. Trace-driven evaluations show that the proposed SSR outperforms state-of-the-art algorithms over various settings, indicating the practicability and robustness of SSR to be deployed in real-world short-form video feed applications.

Acknowledgment

This work is partially supported by National Key Research and Development Program No. 2017YFB0803302, Beijing Academy of Artificial Intelligence (BAAI), and NSFC 61632017.

References

[1] V. Cisco, "Cisco visual networking index: Forecast and trends, 2017–2022 white paper," *Porto Salvo, Lisboa. Disponível em:¡ https://www. cisco. com/c/pt_pt/about/press/news-archive-2018/20181127. html¿, Acesso em*, vol. 17, 2019.

[2] X. Chen, D. Liu, Z.-J. Zha, W. Zhou, Z. Xiong, and Y. Li, "Temporal hierarchical attention at category- and item-level for micro-video click-through prediction," in *MM*, 2018.

[3] X. Zhao, L. Xia, L. Zhang, Z. Ding, D. Yin, and J. Tang, "Deep reinforcement learning for page-wise recommendations," in *RecSys*, 2018, pp. 95–103.

[4] D. Crankshaw, X. Wang, G. Zhou, M. J. Franklin, J. E. Gonzalez, and I. Stoica, "Clipper: A low-latency online prediction serving system," in *NSDI*, 2017, pp. 613–627.

[5] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 2017, pp. 197–210.

[6] Y. Zhang, P. Zhao, K. Bian, Y. Liu, L. Song, and X. Li, "Drl360: 360-degree video streaming with deep reinforcement learning," in *IEEE INFOCOM 2019*, 2019, pp. 1252–1260.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017, pp. 5998–6008.

[8] Y. Chen, Y. Liu, B. Zhang, and W. Zhu, "On distribution of user movie watching time in a large-scale video streaming system," in *2014 IEEE ICC*. IEEE, 2014, pp. 1825–1830.

[9] Y. Shi, M. Larson, and A. Hanjalic, "List-wise learning to rank with matrix factorization for collaborative filtering," in *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010, pp. 269–272.

[10] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*, ser. RecSys '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 191–198. [Online]. Available: https://doi.org/10.1145/2959100.2959190

[11] Q. Chen, H. Zhao, W. Li, P. Huang, and W. Ou, "Behavior sequence transformer for e-commerce recommendation in alibaba," in *Proceedings of the 1st International Workshop on Deep Learning Practice for High-Dimensional Sparse Data*, 2019, pp. 1–4.

[12] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *ICML*, 2016, pp. 1928–1937.

[13] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *ICML*, 2013, pp. 1310–1318.

[14] B. Meixner, J. W. Kleinrouweler, and P. Cesar, "4g/lte channel quality reference signal trace data set," in *MMSys*. ACM, 2018, pp. 387–392.

[15] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck, "HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks," *IEEE Comm. Letters*, pp. 2177–2180, 2016.

[16] X. Chen, D. Liu, C. Lei, R. Li, Z.-J. Zha, and Z. Xiong, "Bert4sessrec: Content-based video relevance prediction with bidirectional encoder representations from transformer," in *ACM MM*, 2019, pp. 2597–2601.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[18] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, "Stamp: short-term attention/memory priority model for session-based recommendation," in *SIGKDD*, 2018, pp. 1831–1839.

[19] B. Hidasi and A. Karatzoglou, "Recurrent neural networks with top-k gains for session-based recommendations," in *CIKM*. ACM, 2018, pp. 843–852.

[20] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, 2017, pp. 1419–1428.