Video Big Data Retrieval Over Media Cloud: A Context-Aware Online Learning Approach

Yinan Feng, Pan Zhou, *Member, IEEE*, Jie Xu, *Member, IEEE*,
Shouling Ji, *Member, IEEE*, Dapeng Wu, *Fellow, IEEE*

***Abstract*—Online video sharing (e.g., via YouTube or YouKu) has emerged as one of the most important services in the current Internet, where billions of videos on the cloud are awaiting exploration. Hence, a personalized video retrieval system is needed to help users find interesting videos from big data content. Two of the main challenges are to process the increasing amount of video big data and resolve the accompanying "cold start" issue efficiently. Another challenge is to satisfy the users' need for personalized retrieval results, of which the accuracy is unknown. In this paper, we formulate the personalized video big data retrieval problem as an interaction between the user and the system via a stochastic process (SP), not just a similarity matching, accuracy (feedback) model of the retrieval; introduce users' real-time context into the retrieval system; and propose a general framework for this problem. By using a novel *contextual* multi-armed bandit-based algorithm to balance the accuracy and efficiency, we propose a context-based online big-data-oriented personalized video retrieval system. This system can support datasets that are dynamically increasing in size and has the property of cross-modal retrieval. Our approach provides accurate retrieval results with *sublinear* regret and *linear* storage complexity and significantly improves the learning speed. Furthermore, by learning for a cluster of similar contexts simultaneously, we can realize sublinear storage complexity with the same regret but slightly poorer performance on the "cold start" issue compared to the previous approach. We validate our theoretical results experimentally on a tremendously large dataset; the results demonstrate that the proposed algorithms outperform existing bandit-based online learning methods in terms of accuracy and efficiency and the adaptation from the bandit framework offers additional benefits.**

***Index Terms*—Big data, video retrieval, online learning, media cloud, contextual bandit, online learning**

I. Introduction

ONLINE video service has proliferated in recent years and emerged as one of the most important services in the current Internet. More than 300 hours of videos are uploaded to the largest online video sharing service website, namely, YouTube[1], every minute, where billions of videos are already stored in a multimedia-aware cloud (media cloud), which can be watched, commented on and shared. Due to the gigantic base and the rapid sustained growth of online videos, the content that is stored in the cloud for users can be considered nearly infinite. However, traditional video retrieval systems [1], [2] cannot handle such a large volume of data. Moreover, videos with similar visual information differ substantially in terms of presentation form and content and users have preferences. This difference motivates us to propose a novel method, which can work efficiently in the big data domain with scalability and provide personalized retrieval results.

Users utilize video web services for a large variety of reasons; two main reasons are to find videos that they had found elsewhere or on a particular topic and to find interesting videos with no specified objective [3]. Hence, the system's goal should be to provide personalized retrievals for various users. To serve these two types of users, our system must have two types of inputs (we call both of them *contexts*): users' *query conditions* and *real-time conditions* (e.g., age, gender, and location). The latter will play an important role, especially when a user's query condition is fuzzy, e.g., when searching "US presidential election", a user who is at the office may want to watch the news, whereas a user who is taking a break may be more interested in animations of presidential candidates' jokes.

The abovementioned problem demonstrates the demand for a *personalized* video big data retrieval algorithm. Existing personalized retrieval methods such as relevance feedback (RF) assume the personalization is only reflected by similarity measures (e.g., color and shape) [4]. We argue that a metric of a personalized retrieval should consider not only videos' similarity but also the overall user satisfaction. However, it is accompanied by the problem that each user's attitude toward a video is unknown and ever-changing. Thus, the accuracy in a personalized retrieval problem is unknown. Thus, in contrast to the crude settings of other approaches, we formulate the personalized video big data retrieval problem as an interaction between the user and the system with an unknown-distribution model. We utilize SP to model the satisfaction because there is no context that can completely describe a person. Two users that have the same context may have different attitudes toward the same video. A model could better handle the personalized retrieval problem.

However, most existing works, even various approaches that were proposed for the large-scale video retrieval problem such as [6], [7], are based on the similarity matching method, and their results are fixed according to the dissimilarity between the query condition and the output. Thus, they will perform poorly if the accuracy of the retrieval result is unknown. In

Yinan Feng and Pan Zhou are from Huazhong University of Science & Technology, Wuhan, Hubei 430072, China. Email: yinanfenghust@gmail.com, panzhou@hust.edu.cn
J. Xu is with the Department of Electrical and Computer Engineering, University of Miami, Coral Gables, FL 33146, USA. Email: jiexu@miami.edu
Shouling Ji is with the College of Computer Science and Technology, Zhejiang University, Hangzhou, Zhejiang 310027, China. Email: sji@gatech.edu
Dapeng Wu is with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA. Email: wu@ece.ufl.edu

[1]www.YouTube.com

addition, the computational cost of calculating the similarity with a high-dimensional feature is very expensive [8], and such features consider little information about user feedback, which should be the most significant performance measure of a system. Hence, these features are suitable for near-duplicate video detection but may not satisfy real users' requirements. Another shortcoming of the stochastic process feedback model is that the accuracy of the retrieval problem for a context can only be learned by observing how satisfied users were with the video when it was retrieved in similar contexts. Therefore, the retrieval system faces the dilemma of exploration versus exploitation: the system must search for a balance between exploration of the most informative videos with highly uncertain performance and exploitation of the most positive videos with the highest estimated performance [5] [35]. *Cold start* is another challenging issue in big data video retrieval systems, which is due to very few or even no prior interaction histories of diverse users and items being available initially. For this multicase problem, new users are frequently visiting videos, and new videos are frequently being uploaded; hence, the "cold start" issue must be tackled. Clustering users in terms of context and videos according to their properties can address this issue and enable the system to exploit information that is obtained from users that have similar contexts and videos that have similar properties.

To process the stochastic results and solve this big data multicase problem, we propose a novel contextual online learning method, namely, a *contextual* multi-armed bandit (MAB)-based algorithm, to replace the similarity matching method as a general framework for the personalized video big data retrieval problem. MAB is one of the simplest and best-performing online learning algorithms [5], [9] due to its finite-time optimality guarantee when facing the exploration-exploitation dilemma. By leveraging the online learning method, the proposed methods can use historical retrieval results to reinforce future retrievals.

Our proposed retrieval system can be divided into three parts: *indexing video*, a *context extractor*, and a *retrieval algorithm*. To balance efficiency and accuracy, the videos are indexed initially as a binary tree, which is called a *cover tree*. The cover tree has a top-down design, and each node is a video cluster. The size of the clusters is subject to various restrictions, and videos are organized according to a weak Lipschitz condition, which is a milder restriction and is similar to the restrictions of most video service webs. Initially, there is only one root node, which contains all video indices, and others are added during the retrieval process. This method is more suitable for massive, dynamically increasing video data. Then, the context extractor extracts users' real-time conditions and combines them with the query as input. One approach is to extract from social networks [10] since users do not fill in much information on video service webs, and most of them support *social login*, which allows new users to sign in with their social network identities, such as their Facebook or Twitter accounts. Since we do not use similarity matching, the proposed methods have the property of cross-modal retrieval, and the query condition can be of any type (e.g., text, image, example, object or sketch). Since the sources of real-time conditions can vary over time, the bandit framework is the optimal choice.

To this end, in contrast to most existing content-based or text-based retrieval algorithms, we propose two novel contextual bandit-based online learning algorithms: the adaptive cover tree algorithm (ACT) and static-adaptive cover tree algorithm (SACT). Our algorithms operate in the time-slot model. At each time slot, the user inputs his context. Then, the system evaluates each video cluster's payoff by exploiting the helpful historical information and selects a video from the selected video cluster that has the highest estimated payoff. After receiving the result, the user will feed back a payoff, namely, the *reward* in bandit problems. This method follows stochastic distributions, reflects users' satisfaction and will be recorded to facilitate future retrievals. Our algorithm is a general framework for the personalized video big data retrieval problem. Hence, our approach can be incorporated with 1) space-partitioning and dimension-reduction methods in the video initialization steps, where original videos are mapped, indexed and partitioned into the video space, cover-tree formulation and context, and 2) similarity measurement algorithms when a video cluster is selected. The main motivation of SACT is as follows: via simultaneous learning for a cluster of similar contexts instead of individual learning for each context, it can substantially reduce the space complexity to sublinear and accelerate the learning process. Thus, by sacrificing a small amount of cold start performance, SACT can help service providers reduce the resource cost of computing and storing in the cloud. The SACT cold start performance loss will be evaluated experimentally.

We use *regret* to measure the performance of our algorithms, which is defined as the difference between the expected total reward the system obtains via the optimal strategy given complete knowledge about the payoff distribution of all online videos for all possible contexts and the expected total reward that can be achieved via our algorithms. In other words, regret is the total loss that is due to the lack of knowledge about the payoff distribution. Then, we demonstrate that the proposed algorithms achieve sublinear regret in the number of users that have arrived thus far. Hence, for each possible context, the best video to select can be learned.

The main contributions of this paper are summarized as follows:

- We formulate the personalized video big data retrieval problem as a stochastic-process feedback model, which models an interaction between the user and the system.
- We propose a general framework for the personalized video big data retrieval problem, namely, a novel context-aware online learning algorithm for real large datasets, which are ever-increasing in size. This framework achieves a sublinear regret bound; thus, it is an optimal learning strategy over any specified finite time interval.
- We take into account users' real-time conditions in the retrieval process to realize personalized retrieval.
- Our approach does not require similarity matching. Hence, it has the property of cross-modal retrieval.

- The space complexity is linear for ACT and sublinear for SACT. Hence, our algorithms will only slightly increase the storage burden compared to the cloud; thus, they are highly suitable for real-time big data applications.
- We also address the issue of cold start via contextualization and the bandit's constant exploration in the learning process.

The remainder of the paper is organized as follows. In Section II, we describe the related work. In Section III, we formalize the context-aware online video retrieval problem and present the system model. In Sections IV and V, we present our ACT and SACT algorithms and analyze the regret bounds. A complexity analysis is presented in Section VI. The experimental design and simulation result discussion are provided in Section VII. Section VIII presents the conclusions of the paper.

II. Related Work

In recent years, multimedia applications and services have emerged as a significant part of the Internet, with intense demands for cloud computing. A media cloud focuses on how the cloud can provide quality of service (QoS) provisioning or, specifically, raw resources, such as hard disk, CPU, and GPU, which are rented by media service providers (MSPs) to serve users [11], which is the basis of multimedia retrieval over the cloud.

Video indexing and retrieval have a wide range of applications and motivate researchers worldwide. An overview of the process of a video indexing and retrieval framework is presented in [12]. The whole process can be divided into two parts: first, extracting feature and indexing videos and, second, query and relevance feedback. According to the application, features can be extracted from the video database. Static features of keyframes [13], [14], object features [15], [16], and motion features [17], [18] are the three most common types of features.

By using features, a video should be indexed to facilitate video retrieval. Recently, learning-based hashing methods have become popular for indexing and approximate nearest neighbor search of large-scale media data [6], [20], [21]. J. Wang *et al.* present a survey of learning-based hash algorithms and categorize them according to their similarity-preservation properties into pairwise similarity preserving, multiwise similarity preserving, implicit similarity preserving, quantization, and end-to-end hash learning, in which the hash codes are computed directly from the original object [19]. When hash functions are learned, only the XOR operation and the bit count operation on hash codes are needed to compute the similarity between two videos. However, to generate fixed-length hash codes, determining keyframes in videos is important but formidable. Moreover, a hashing algorithm must store all videos hash codes to measure the similarity by summing the pairwise similarity between frames' hash codes, which leads to high storage and computational consumption [8], whereas our method only requires similar videos to be stored together and may not require the details of features' values. Other methods such as [22] jointly exploit the feature relationships and the class relationships, which might not be able to handle big data problems. For state-of-the-art works, in [23] and its improved version, namely, [24], the authors aim at improving video hashing by taking temporal information into consideration. Moreover, in [25], the authors propose a general framework for incorporating quantization-based methods into the conventional property-preserving methods. These approaches perform outstandingly as similarity measurement methods; however, they have the inherent defects of similarity measures in personalized retrieval, as mentioned above.

Once the video index has been obtained, the video retrieval process can begin. According to the query type, a query can be classified, e.g., as query by examples, query by objects, or query by keywords. Near-duplicate video retrieval (NDVR) [26], [27] is an application of query by examples that has been frequently utilized in recent years. Another benefit of not using similarity matching is that our method is applicable to queries of any type if the query condition is being mapped into the context space. Like the cross-modal hash algorithm that was proposed in [28], our proposed algorithms have cross-modal retrieval ability.

Substantial work has been carried out on extracting query features. For example, in [29], the authors proposed a method of query by sketch, where trajectories that are drawn by users are extracted. In [1], the authors normalized both the descriptions of concepts and the query text and computed the similarity with the text by using a vector space model. Connotations from movie scenes can be extracted and validated to facilitate retrieval [30], [31]. For users' real-time conditions, since video service websites also support social login, we can learn correlated users' features from social networks [10]. Meanwhile, both users' emotions and emotion preferences can be used as context dimensions to facilitate retrieval [30], [31]. Similar to our starting point, H. Ghosh *et al.* and D.Vallet *et al.* use users' feedback to provide personalized retrieval output in [32], [33]. Although their work slightly improves the retrieval result, it is still based on similarity rather than satisfaction.

One of the most relevant prior works in online learning is the RF method. Given users' feedback, the key step in an RF scheme is the selection of a subset of image features for constructing a suitable dissimilarity measure and effectively modeling high-level concepts and user perception subjectivity [4], [34]. Y. Rui *et al.* introduce this method into content-based image retrieval (CBIR) [4]. X. Zhou *et al.* provide a review [35], and an evaluation is provided by [36]. Many RF schemes have been proposed for CBIR [34], [37]- [40]; however, content-based video retrieval has not received sufficient attention, and few RF methods that can be applied to video retrieval exist. R. Yan *et al.* propose utilizing pseudo-relevance feedback from retrieved items that are not similar to the query to improve the retrieval performance but not the personalized performance [41]. Shao *et al.* incorporate spatiotemporal localization and use a relevance feedback algorithm to enhance the video retrieval performance [42]. I. Mironică *et al.* propose a novel framework for relevance feedback that is based on the Fisher kernel (FK) for video retrieval [43]. No RF scheme has been proposed for big data problems. The primary difference between our approach and

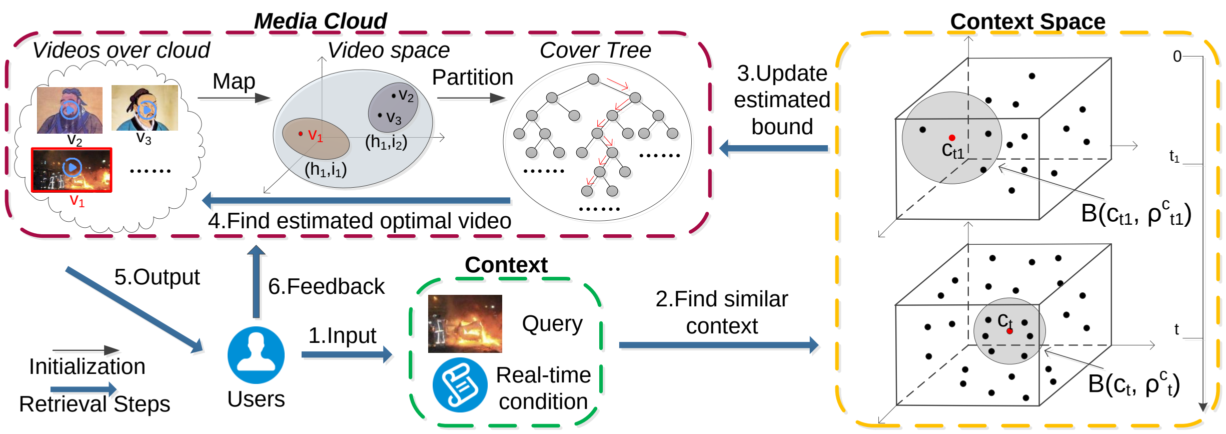

Fig. 1: Workflow of the retrieval system over a media cloud.

RF methods is that we assume the result of a personalized video retrieval to be a stochastic process with an unknown distribution, whereas RF assumes the personalization is only reflected by the similarity measures. We argue that an unknown stochastic process feedback model is more general and more suitable for practical applications of the online video service. Thus, we use users' feedback to modify the weights of subspaces of videos rather than features. Moreover, similar to most existing information retrieval systems, conventional RF schemes that track feedback by a user base their decisions solely on queries and often ignore the search context [50]. This basis typically causes the small sample size problem. To tackle this problem, in our approach, each user is described by his/her context and utilizes other users' historical results. Via a contextual bandit online learning algorithm, our method can balance the selection of most informative (exploration) and most positive (exploitation) images.

Cold start problems, which are due to our exploration of historical results rather than a similarity measure to facilitate retrieval, are divided into three types: (a) recommendations by new users, (b) recommendations for new items, and (c) recommendations for new items by new users [44]. Various state-of-the-art methods address user-side problems by incorporating neighbors of new users [44], [45] or introducing external social information [46]. For the item side, [47] incorporates additional sources of information about both the users and items, and [48] uses a matrix factorization technique to address the problem. [49] aims at proposing a general framework for dealing with both user-side and item-side cold start problems simultaneously with an efficient matrix factorization approach using similarity information. The context-based method is another important technique for dealing with cold start [47], [50], [51], which has benefits in terms of both cold start and accuracy.

The result of personalized video retrieval is typically a stochastic process. We use an online learning method instead of hashing or another offline similarity matching approach. In MAB algorithms, each optional item is named an *arm*, and algorithms estimate the mean reward of each item based on the historical results. By modeling the personalized user features as contexts, the previous contextual MAB that can deal with the multiclass classification problem has been used for various applications, such as image mining [52], traffic prediction [53], and recommendation systems [54]- [51]. However, these works require the traversal of all items (*arms*) before the algorithm is run, which is difficult to achieve in big data scenarios. Works such as [51] used a bottom-up design to build an item tree from the leaf to the root such that they must know the number of items in advance. [56] and [57] proposed algorithms for continuous arm space for the optimization problem that are unlike traditional bandit methods. However, personalized retrieval is a multiclass classification problem, and the multicase scenario should be considered. Additionally, by taking users' real-time conditions into account, we substantially improve the video retrieval performance for the first time.

III. Problem Formulation

In this section, we formulate the personalized video big data retrieval problem, which is an interaction between users and a retrieval system. First, we describe the system model and important elements. Then, we explain our stochastic process feedback model in detail.

A. System Model

We illustrate the workflow of the video retrieval system over the media cloud in Fig. 1. In *initialization*, the set of videos, which is denoted as $\mathcal{V} = \{v_1, v_2, \cdots\}$, is mapped into a d_V-dimensional measurable space $(\mathcal{V}, l)$, where each video is described as a d_V-dimensional feature vector and l: $\mathcal{V}^2 \to \mathbb{R}$ is the video dissimilarity function. Each dimension of the video feature vector represents one feature of the video. These features can be text, semantic and high-level features (HLFs), such as video type, tag, duration, connotation, motion, objects, and phenomena. The function l satisfies $l(v, v') \geq 0$ for all $(v, v') \in \mathcal{V}^2$ and $l(v, v) = 0$. Typically, similar videos (e.g., v_2 and v_3) have a small function value and are close in the space, and dissimilar videos (e.g., v_1 and v_2) are distant from each other. As discussed previously, this space will be partitioned and organized into the *cover tree*, which is denoted as $\mathcal{T}$, as our retrieval index. The tree is stored in the media cloud with all videos, built and updated dynamically during the retrieval process. The main focus of this work is the retrieval steps (represented by blue arrows): if the dissimilarity function and the video space index are given, the system learns to retrieve videos that are searched by users.

The users' context set is denoted by $\mathcal{C} = \{c_1, c_2, \cdots\}$. The system works in a time slot. Let u_t and $c_t \in \mathcal{C}$ be the user and context that arrive at time $t = 1, 2, 3, \cdots$, and let v_t and $r_t \in [0, 1]$ be the corresponding retrieval result and received reward, respectively. $\mathcal{T}_t$ denotes the cover tree that we built at time t. For any $t > 0$, the history space $\mathcal{H}_t := ([0, 1] \times \mathcal{V} \times \mathcal{C})^t$ is defined as the space of past rewards, retrieval results and contexts before t, where $\mathcal{H}_0 = \emptyset$. Specifically, this historical information is our main foundation for retrieval. As illustrated in Fig. 1, at each time t, the following events occur sequentially: 1) A user u_t sends c_t, which denotes his/her retrieval request (e.g., a photo of a fire disaster), together with the real-time condition, to the online retrieval system. 2) Our system searches $\mathcal{H}_t$ to find useful historical results that have a similar context to c_t. 3) Our retrieval algorithms calculate and estimate the expected reward for each video cluster according to these historical results.4) The leaf node with the highest estimated reward in $\mathcal{T}_t$ will be selected (e.g., h_1, i_1 in Fig. 1). 5) A video v_t that belongs to that node, e.g., video v_1 of firefighting in our example, will be the output. 6) Once the user has watched this video, he/she returns a reward r_t to reflect the attitude toward v_t, which will be recorded in $\mathcal{H}_t$ to facilitate future retrieval. Finally, $\mathcal{T}_t$ explores new nodes if the size of an existing node is too large. All these operations are performed in and the records are stored in the cloud; hence, users' computing and storage resources are not consumed.

B. Context Space Model

We model the d_C-dimensional measurable context space as $(\mathcal{C}, s_C)$, which includes the context set $\mathcal{C}$ and distance function s_C: $\mathcal{C}^2 \to [0, \infty)$. Each context $c \in \mathcal{C}$ is a d_C-dimensional vector, which includes a d_C^q-dimensional query condition and a d_C^r-dimensional real-time condition of the user. For simplicity of exposition, we assume that $\mathcal{C} = [0, 1]^{d_C}$. $\mathcal{C}$ only describes users' context set but not the tracking of a user. s_C is utilized to determine the distance between any two contexts such that $s_C(c, c') \geq 0$ for all $(c, c') \in \mathcal{C}^2$ and $s_C(c, c) = 0$. In Euclidean space $\mathcal{C}$, the function s_C can be the Euclidean norm or any other norm. For any measurable space (e.g., $(\mathcal{V}, l)$ or $(\mathcal{C}, s_C)$), once the element set $\mathcal{S}$ and the distance function f have been specified, we denote the maximum dissimilarity of two elements in a subset $A \subseteq \mathcal{S}$ as $diam(A) := \sup_{x,y \in A} f(x, y)$. In addition, $\mathcal{B}(s, \rho) := \{s' \in \mathcal{S} : f(s, s') \leq \rho\}$ denotes the subset that contains all similar elements whose distance to $s \in \mathcal{S}$ is no longer than a constant $\rho > 0$. Since its definition coincides with the geometrical concept of a sphere, for the context space, we define $\mathcal{B}(c_t, \rho)$ as a s_C-ball with radius $\rho > 0$ and center c_t to identify useful historical results of current retrieval approaches. Examples are shown in the right part of Fig. 1.

C. Video Cover Tree

Since millions of content providers are uploading videos on the Internet, $\mathcal{V}$ will expand over time, and our model is based on the assumption that the massive number of videos can be viewed as nearly infinite. Thus, we use the binary tree $\mathcal{T}$ to reduce the possible options for retrieval, in which each node (h, i) is a video cluster at depth h and index i among the nodes at the same depth and stores the estimated expected reward of the corresponding video cluster. The area $\mathcal{P}_{h,i} \subset \mathcal{V}$ corresponds to the video subset in the node (h, i). The root node is indexed by $(0, 1)$ and satisfies $\mathcal{P}_{0,1} = \mathcal{V}$. The two child nodes of (h, i) are denoted by $(h + 1, 2i - 1)$ and $(h + 1, 2i)$. Since it is a binary tree, for any $h \geq 0$ and $1 \leq i, j \leq 2^h$, the following holds: $\mathcal{P}_{h,i} \cap \mathcal{P}_{h,j} = \emptyset$ and $\mathcal{P}_{h,i} = \mathcal{P}_{h+1,2i-1} \cup \mathcal{P}_{h+1,2i}$.

Similar to the above context, we call $\mathcal{B}(v, \rho)$ an l-ball with radius $\rho > 0$ and center $v \in \mathcal{V}$, which will be used to restrict the minimum size of video clusters. Since the cover tree is our retrieval index, the subset partition should not be arbitrary. A subset that is too large (i.e., videos in the subset differ substantially) is useless, as it cannot contain any valuable information to facilitate retrieval. Meanwhile, a subset that is too small may cause overfitting phenomena. Thus, both the maximum and minimum sizes of each node should be restricted as follows:

Assumption 1 (Cover tree structure). *There exist $\nu_1, \nu_2 > 0$ and $0 < \rho < 1$ such that for any nodes $(h, i), (h, j) \in \mathcal{T}$:*

(a) $\quad diam(\mathcal{P}_{h,i}) \leq \nu_1 \rho^h$.

(b) $\quad \exists v_{h,i}^o \in \mathcal{P}_{h,i}$ *s.t.* $\mathcal{B}_{h,i} := \mathcal{B}(v_{h,i}^o, \nu_2 \rho^h) \subset \mathcal{P}_{h,i}$.

A cover tree that satisfies this assumption can be used as an index and is both efficient and accurate. This assumption is not the natural characteristics of the video space but only the requirements of the format of the input data. Constants ν_1, ν_2, ρ depend on dissimilarity function l and the original data characteristics. In practice, ρ is often approximately 0.5. We illustrate an example in Fig. 1. Two similar videos, namely, v_2 and v_3, in the same node should satisfy $l(v_2, v_3) \leq \nu_1 \rho^{h_1}$. In addition, $diam(\mathcal{P}_{h_1,i_1}), diam(\mathcal{P}_{h_1,i_2}) \geq \nu_2 \rho^{h_1}$. Moreover, our approach is specialized for big data scenarios. Hence, even after running for a long time, leaf nodes will not merely contain one item that cannot be divided.

D. Regret Analysis

We use the notation $r(v, c) \in [0, 1]$ to relate reward r to browsed video v and users' context c (particularly, $r(v, c) = 0$ means the user does not watch the video). The basic assumption is that the randomness of the reward mainly comes from users' diverse mental conditions and a video with a specified context will lead to a corresponding distribution. Thus, we assume the distributions of r are independent and identically distributed (i.i.d.), which only depend on v and c. Hence, we use $\mu_{v,c} := \mathbb{E}\{r(v, c)\}$ to represent the expected reward that can be obtained from video v with context c. The reward has two forms: explicit and implicit. Explicit feedback asks users to actively return a measurable result (e.g., score or rating), and implicit feedback is observed from the user's diverse actions and reflects the user's attitude toward the video (e.g., watching the video or not, or watching duration). Both can reflect user satisfaction [12].

Algorithm 1 Adaptive Cover Tree Algorithm

```
1: Input : Parameters $\nu_1 > 0$, $\rho \in (0,1)$, $c > 0$, confidence $\delta \in (0,1)$, and cover tree structure $(\mathcal{P}_{h,i})_{h\geq 0, 1\leq i\leq 2^h}$.
2: Initialize : $t = 1$, $\mathcal{T}_t = \{(0,1),(1,1),(1,2)\}$, $H(t) = 1$, $U_{1,1}(t) = U_{1,2}(t) = C_{max}$.
3: loop
4:     The user inputs the current context $c_t$; find its $\rho_t^c$-neighbor $B(c_t, \rho_t^c)$.
5:     for all video clusters $(h,i) \in \mathcal{T}_t$ backward from the leaf nodes $(H(t), i)_{1\leq i\leq 2^{H(t)}}$ do
6:         Update the estimated node's upper bound $U_{h,i}(t)$ in Eq. (2).
7:         Update the tighter retrieval reward bound $B_{h,i}(t)$ in Eq. (3).
8:     end for
9:     $(h_t, i_t), P_t \leftarrow Optpath(\mathcal{T}_t)$.
10:    The user browses the video that corresponds to $v_{h_t,i_t}$ and gives his/her feedback.
11:    Go to the next time slot: $t = t + 1$.
12:    if $T_{h_t,i_t}(t) \geq \tau_{h_t}(t)$  $AND$  $(h_t, i_t) \in leaf(\mathcal{T}_t)$ then
13:        $\mathcal{T}_t \leftarrow \mathcal{T}_t \bigcup \{(h_t + 1, 2i_t - 1), (h_t + 1, 2i_t)\}$.
14:        $U_{h_t+1,2i_t-1}(t) = U_{h_t+1,2i_t}(t) = C_{max}$.
15:    end if
16: end loop
```

Algorithm 2 The *Optpath* function

```
1: Input : Tree $\mathcal{T}_t$.
2: Initialize : $(h,i) \leftarrow (0,1)$, $P \leftarrow (0,1)$, $T_{0,1}(t) = \tau_0(t) = 1$.
3: while $T_{h,i}(t) \geq \tau_h(t)$  $AND$  $(h,i) \notin leaf(\mathcal{T})$ do
4:     if $B_{h+1,2i-1} \geq B_{h+1,2i}$ then
5:         $(h,i) \leftarrow (h+1, 2i-1)$.
6:     else
7:         $(h,i) \leftarrow (h+1, 2i)$.
8:     end if
9:     $P \leftarrow P \bigcup \{(h,i)\}$.
10: end while
11: Output : $(h,i)$ and $P$.
```

For each cluster (h,i), we randomly select a video $v_{h,i} \in \mathcal{P}_{h,i}$ to represent the video cluster. At each time t, selecting the node (h_t, i_t), we output the corresponding video v_{h_t,i_t} as the result to the user. Using a fixed option, it is possible to bound the worst case in the theoretical analysis. The choice of $v_{h,i}$ changes at every round in experiments and practical applications. Assuming the maximizer $v_t^* = \text{argmax}_v \mu_{v,c_t}$ exists, we denote the corresponding maximum $\mu_{v_t^*,c_t}$ by μ_t^*. Each video has a similar impact for similar users, and each user has a similar reflection on similar videos. This assumption is formalized as the widely adopted Lipschitz condition assumption.

Assumption 2 (Lipschitz condition). *Given two contexts $c, c' \in \mathcal{C}$, for each video $v \in \mathcal{V}$ and $t > 0$, the following hold: $|\mu_{v,c} - \mu_{v,c'}| \leq L_C s_C(c,c')$ and $\mu_t^* - \mu_{v,c_t} \leq l(v_t^*, v)$, where L_C is the Lipschitz constant in the context space.*

The Lipschitz condition for contexts and the Lipschitz constant L_C are only required for theoretical analysis and need not be known. Meanwhile, the Lipschitz condition for the video space only requires the expected reward function to be Lipschitz with respect to the maximum value. This condition is weaker than the standard condition and is called the *local smoothness* assumption.

To measure the accuracy of the system at step t, we denote the regret at t by $\Delta_t = \mu_t^* - r_t$. Over n steps, the expectation regret, which is denoted as R_n, is defined as

$$R_n = \mathbb{E}\sum_{t=1}^{n}\Delta_t = \sum_{t=1}^{n}(\mu_t^* - \mathbb{E}r_t) = \sum_{t=1}^{n}(\mu_t^* - \mu_{h_t,i_t}(t)), \quad (1)$$

where $\mu_{h_t,i_t}(t)$ is a short notation for $\mu_{v_{h_t,i_t},c_t}$. The main objective of the system is to choose a strategy that minimizes the total regret R_n. A sublinear regret means the algorithm can converge to the optimal strategy, since $lim_{n\to\infty} R_n/n \to 0$.

In addition, we require a tighter upper bound for a near-optimal content subset in the theoretical analysis. We define a *near-optimal video* as follows. Let $\epsilon > 0$. The subset of ϵ-optimal videos is defined as $\mathcal{V}_\epsilon = \{v \in \mathcal{V} : \mu_t^* - \mu_{v,c_t} \leq \epsilon\}$. In the next section, we prove that our approaches always output near-optimal videos with high probability. Then, we characterize the *scale* of the problem, which refers to how *large* the set of ϵ-optimal videos in $\mathcal{V}$ is, using the concept of *packing number*. For $\epsilon' < \epsilon$, there exists a *packing constant* C_V such that $\mathcal{N}(\mathcal{V}_\epsilon, l, \epsilon') \leq C_V(\epsilon')^{-d_V}$, where $\mathcal{N}(\mathcal{V}_\epsilon, l, \epsilon')$ is the maximum number of sphere areas with radius ϵ' that are disjoint with one another in the region $\mathcal{V}_\epsilon$ with respect to the distance measure l. Similarly, we denote the minimal number of sphere areas with radius $\rho > 0$ that can cover the context space $\mathcal{C}$, with respect to the distance measure s_C by $\mathcal{N}_\rho(\mathcal{C})$. In addition, $\mathcal{N}_\rho(\mathcal{C}) \leq C_C\rho^{-d_C}$, where C_C is the *covering constant* of $\mathcal{C}$.

TABLE I: Notation

c_t	The context at time t.
v_t	The retrieval result at time t.
r_t	The received reward at time t.
(h_t, i_t)	The cluster that is selected by ACT at time t.
ν_1	The maximum distance in the video space.
$\mathcal{P}_{h,i}$	The corresponding video subset of node (h,i).
$\mathcal{T}_t$	The cover tree that has been already built at time t.
$H(t)$	The depth of tree $\mathcal{T}_t$.
P_t	The traversal path from the root node to node (h_t, i_t).
C_{max}	A maximum number, which is used for programming environment supports.
$B(c_t, \rho_t^c)$	An s_C-ball with radius ρ_t^c and center c_t.
$\mathcal{T}(B(c_t, \rho_t^c))$	The set of past context arrival times in $B(c_t, \rho_t^c)$.
$T_{h,i}(t)$	The number of times the cluster (h,i) has been retrieved in $B(c_t, \rho_t^c)$.
$\mathbb{I}$	The indicator function.
$\widehat{\mu}_{h,i}(t)$	The empirical reward of cluster (h,i) at time t.
$U_{h,i}(t)$	The estimated reward upper bound of node (h,i) at time t.
$\tilde{\delta}(t)$	$min\{c_1\delta/t, 1\}$ ($c_1, \delta > 0$ are constants).
t^+	$2^{\lfloor log(t)\rfloor+1}$.
$B_{h,i}(t)$	The tighter reward bound of node (h,i) at time t.
$\tau_h(t)$	The threshold of selected times for each node.

IV. Adaptive Cover Tree Algorithm

In this section, we present our ACT algorithm, which utilizes historical information to facilitate searching for the optimal retrieval result. In addition, we prove that it achieves a sublinear learning regret.

A. Algorithm Description

The details of the ACT are shown in Alg. 1. Table I summarizes the notations that we use in the description and analysis of our algorithm. The algorithm consists of four main steps: 1) obtaining input and historical information (Line 4); 2) updating (Line 5-8); 3) searching for the result and outputting it (Line 9-11); and 4) expansion (Line 12-15).

1) Obtaining input and historical information: At each time slot t, the system receives a query with the user's context, which is denoted as c_t. To facilitate the retrieval process, ACT utilizes historical retrieval results that have similar context to c_t. To formally define the meaning of "similar", we utilize the concept of s_C-ball, which is defined in Section II. Specifically, we consider a relevant ball, namely, $B(c_t, \rho_t^c)$, in the context space, whose center is c_t and radius is $\rho_t^c = min\{1, (\log t/t)^\alpha\}$ $(0 < \alpha < 1)$, as shown in the right part of Fig. 1. Via this operation, we select historical data that are similar to the current scenario and call these historical data c_t's ρ_t^c-neighbors. The size of the ρ_t^c-neighbors decreases with time, whereas the retrieval accuracy increases. The set of past context arrival times in the ball $B(c_t, \rho_t^c)$ is denoted by

$$\mathcal{T}(B(c_t, \rho_t^c)) = \{\tau : \tau < t, c_\tau \in B(c_t, \rho_t^c)\}.$$

2) Updating: In the next step of the retrieval process, the ACT updates the estimated reward of the whole cover tree based on the ρ_t^c-neighbor. First, we define two useful variables: The number of times that video cluster (h, i) has been retrieved in the ball is denoted by

$$T_{h,i}(t) = \sum_{\tau \in \mathcal{T}(B(c_t, \rho_t^c))} \mathbb{I}\{h_\tau = h, i_\tau = i\}.$$

$T_{h,i}(t)$ is the total number of times that cluster (h, i) has been retrieved when past users have similar context to c_t. Then, the empirical retrieval reward $\widehat{\mu}_{h,i}(t)$ of $v_{h,i}$ is computed as

$$\widehat{\mu}_{h,i}(t) = \frac{1}{T_{h,i}(t)} \sum_{\tau \in \mathcal{T}(B(c_t, \rho_t^c))} r_\tau.$$

This reward is the sample-average reward in the relevant ball. Since each node in the tree not only includes $v_{h,i}$ but also covers a partition of all videos ($\mathcal{P}_{h,i}$), ACT computes an upper bound, which is denoted as $U_{h,i}(t)$, to represent the estimated reward upper bound of node (h, i). For each node $(h, i) \in \mathcal{T}_t$, the upper bound $U_{h,i}(t)$ is computed as

$$U_{h,i}(t) = \widehat{\mu}_{h,i}(t) + \nu_1 \rho^h + \sqrt{\frac{c^2 log(\frac{1}{\bar{\delta}(t^+)})}{T_{h,i}(t)}}, \tag{2}$$

The third term represents the uncertainty of $\widehat{\mu}_{h,i}(t)$ in estimating $\mu_{h,i}(t)$. The sum of these two terms accounts for the upper confidence bound of $\mu_{v,c}$ [5], where the second term is the maximum size of the node. Thus, Eq. (2) reflects the upper bound of the whole node. However, the ACT algorithm relies on the binary tree $\mathcal{T}$, and $U_{h,i}(t)$ cannot reflect the relation between node (h, i) and its descendants, which makes our estimated upper bound loose. Therefore, the tighter retrieval reward bound, which is denoted as $B_{h,i}(t)$, is designed to associate the parent node and child nodes with a tighter upper bound. More precisely,

$$B_{h,i}(t) = \begin{cases} U_{h,i}(t), & (h,i) \in leaf(\mathcal{T}_t) \\ C_{max}, & T_{h,i}(t) = 0 \\ min[U_{h,i}(t), \max\limits_{j \in \{2i-1, 2i\}} B_{h+1,j}(t)], & otherwise. \end{cases} \tag{3}$$

Importantly, as expressed in Eq. (3), to determine cluster (h, i)'s B-value, we must calculate its children's B-values first. Hence, the whole updating process must begin from the leaf nodes, namely, $(H(t), i)_{1 \le i \le 2^{H(t)}}$, of tree $\mathcal{T}_t$ and proceed backward to the root, namely, $(0, 1)$. Furthermore, because c_t may correspond to a different value of $t > 0$ and the ρ_t^c-neighbor contains diverse contexts at each step, the updating process must be repeated at each time slot, which increases the running time of ACT.

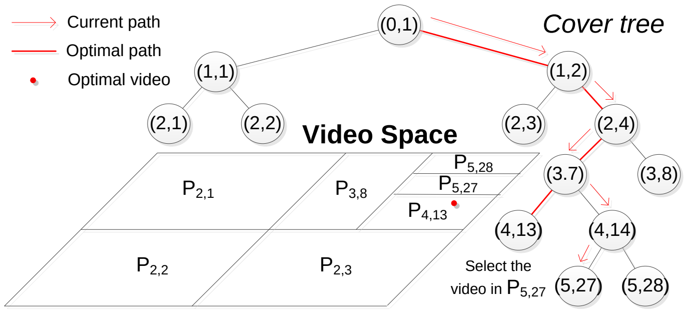

Fig. 2: Example of how to partition the video space and select a node.

3) Searching for the result and outputting it: To identify the most similar video that the user retrieves, ACT calls the *Optpath* function (Alg. 2) to select the child node that has the maximum B-value from the root. Then, it obtains a traversal path, which is denoted as P_t, and stops at (h_t, i_t) (which denotes the cluster that is selected by ACT at time t), which is either a leaf node or a cluster that has not been retrieved sufficiently with respect to a threshold, namely, $\tau_h(t)$, with context c_t. The threshold $\tau_h(t) = c^2 log(\frac{1}{\bar{\delta}(t^+)})/(\rho^h \nu_1)^2$ is used to reduce the depth of the tree and guarantee that each node is exploited sufficiently. The choice of this threshold will be discussed in the next step.

Then, video v_{h_t, i_t} is output to the user. After the user browses it, ACT obtains the reward and stores a new record of (c_t, v_t, r_t). An example of how the *cover tree* partitions $\mathcal{V}$ and how ACT searches for a video is shown in Fig. 2. With current context c_t, the optimal video, which is denoted as v_t^*, is in $\mathcal{P}_{4,13}$. Our ACT calculates the B-values for all nodes, selects $\mathcal{P}_{5,27}$ as output and obtains the path from $\mathcal{P}_{0,1}$ to $\mathcal{P}_{5,27}$.

4) Expansion: In Eq. (2), the last two terms represent the uncertainty of the estimated reward, whereas the second term represents the largest possible difference between two videos in the same cluster, and the third term decreases with the number of retrievals. When videos in a leaf node have been retrieved sufficiently, we expand it and add its children to $\mathcal{T}_t$. Specifically, when the third term is smaller than the second term, the uncertainty is dominated by the node's size. For more accurate retrieval, we must partition the cluster further, expand the tree and set the upper bound U to C_{max} so that it will be enforced for the newly expanded nodes in the next step. These occur when $T_{h,i}(t) \ge \tau_h(t)$, where

$$\nu_1 \rho^h = c\sqrt{\frac{\log\left[\frac{1}{\bar{\delta}(t^+)}\right]}{\tau_h(t)}} \Rightarrow \tau_h(t) = c^2 \frac{\rho^{-2h} log(\frac{1}{\bar{\delta}(t^+)})}{\nu_1^2}. \tag{4}$$

This dynamic expansion process is a top-down process and only depends on the range of each nodes contents; hence, new nodes can be added to the *video space* with no effect on the cover tree. ACT performs well for big data problems, and even after running for a long time, no leaf node will contain only one video. With the above expansion rule, we present a lemma that bounds the maximum depth of the cover tree, which makes the retrieval much more efficient.

Lemma 1. *Given the retrieval threshold $\tau_h(t)$ in Eq. (4), the depth of tree $\mathcal{T}_n$ can be bounded as*

$$H(n) \leq \frac{1}{2(1-\rho)} log(\frac{n\nu_1^2}{2(c\rho)^2}).$$

Proof: See the detailed proof in [59]. ■

Lemma 1 proves that the cover tree is expanded at speed $\mathcal{O}(\log n)$, which guarantees both the retrieval accuracy and sublinear cost for traversing the tree. A threshold that is too large or too small can decrease the accuracy, and a smaller threshold may lead to a higher cost of exploring a suboptimal bunch and a longer traversal path.

B. Regret Analysis

To bound the total regret, we first prove that the average rewards of all expanded nodes are within a confidence interval of empirical estimates with high probability.

Lemma 2. *We define the set of all nodes that are possible in trees with a depth that does not exceed the maximum depth $H_{max}(t)$ as $\mathcal{N}_t = \bigcup_{\mathcal{T}:Depth(\mathcal{T})\leq H_{\max}(t)} Nodes(\mathcal{T})$. A high-probability event is defined as*

$$\varepsilon_t = \{\forall (h,i) \in \mathcal{N}_t, \forall T_{h,i}(t) = 1 \ldots t : \\ |\widehat{\mu}_{h,i}(t) - \mathbb{E}\widehat{\mu}_{h,i}(t)| \leq c\sqrt{\frac{\log\left[1/\widetilde{\delta}(t)\right]}{T_{h,i}(t)}}\}.$$

If $c = 2\sqrt{1/1-\rho}$ and $\widetilde{\delta}(t) = \delta\sqrt[8]{\rho/3\nu_1}/t$, the event ε_t occurs with a probability of at least $1-\delta/t^6$.

Proof: See the detailed proof in [59]. ■

By Lemma 2, we bound the gap between the estimated reward and the expected reward of the suboptimal nodes. Using the above two lemmas, we obtain the following:

Theorem 1 (Regret Bound of ACT). *Let users' feedback be i.i.d., suppose assumptions 1 and 2 hold at each time slot t, and let $\alpha = 1/(d_V + d_C + 2)$. Under the same condition as in Lemma 2, the expectation regret of ACT R_n up to time n is*

$$R_n \leq \frac{6\delta}{5} + 6L_c m + \frac{6\nu_1}{\rho} n^{\frac{d_V+d_C+1}{d_V+d_C+2}} (\log n)^{\frac{1}{d_V+d_C+2}} \\ + \frac{6L_c(d_V+d_C+2)}{d_V+d_C+1} n^{\frac{d_V+d_C+1}{d_V+d_C+2}} (\log n)^{\frac{1}{d_V+d_C+2}} \\ + \frac{96 C_V C_C \rho^{d_V-1}}{(1-\rho^{(1+d_V)})\nu_1 {\nu_2}^{d_V}(1-\rho)} \log(\frac{\sqrt[8]{3\nu_1}n}{\delta\sqrt[8]{\rho}})(\frac{n}{\log n})^{\frac{d_V+d_C+1}{d_V+d_C+2}}.$$

Proof: See the detailed proof in [59]. ■

By our regret definition, Theorem 1 bounds the gap between the optimal choice and the expected feedback. According to the proof, this bound holds with probability $1-\delta$. However, a higher δ can lead to a lower regret bound. Thus, we empirically set $\delta = 0.05$ to guarantee the performance of our algorithm as we did in our experiments. To ensure that the mean reward for all expanded nodes is within the confidence interval with high probability, we set the tradeoff parameter $c = 2\sqrt{1/(1-\rho)}$ via Lemma 2. Users can change this parameter, which will increase $R_n^{\varepsilon^c}$. Moreover, Theorem 1 requires the total retrieval times $n \geq c_1\delta/e = \delta\sqrt[8]{\rho/3\nu_1}/e$. However, parameters δ and ρ are smaller than 1, and ν_1 is always large. Hence, this lemma easily satisfies the requirement.

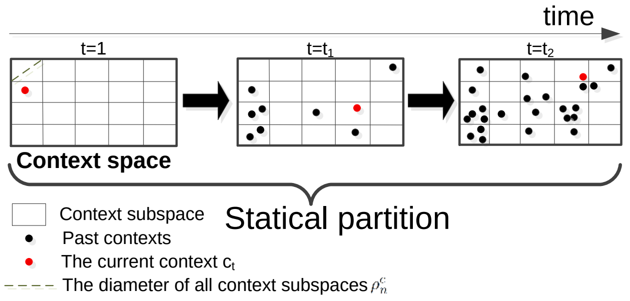

Fig. 3: Illustration of an SACT context partition.

ν_1, which represents the maximum dissimilarity of any two videos in the dataset, will be smaller when videos are organized more regularly. Formally, $\nu_1 = \sup_{v_1,v_2\in\mathcal{V}} f(v_1, v_2)$. According to the choice of $\tau_h(t)$ and Lemma 1, a larger ν_1 leads to a deeper tree. Moreover, the result of step 2 in the proof that $\mu_t^* - \mu_{h_t,i_t}(t) \leq 4\nu_1\rho^{h_t^p} + 2L_C\rho_t^c$ demonstrates that in every step, the regret increases with ν_1. Hence, a more regular video index or, in other words, a lower rate of change of the expected reward function can improve the accuracy and efficiency.

We have proven that Algorithm 1 achieves sublinear regret $R_n = \mathcal{O}(n^{\frac{d_V+d_C+1}{d_V+d_C+2}}(\log n)^{\frac{1}{d_V+d_C+2}})$, which guarantees convergence in terms of the average reward. If d_V or d_C goes to infinity, the regret approaches linearity. This result is because as the dimension of the video increases, the number of nodes that we must explore increases, especially in the first few layers, where we limit the increase in the number of nodes by the tree's structure and the maximum depth in Lemma 1. Similarly, as the dimension of the context increases, the ρ_n^c-covering number of $\mathcal{C}$ increases, and ρ_n^c restricts the covering number growth.

V. Static Adaptive Cover Tree Algorithm

In the previous section, we proposed a contextual query method, where we search similar query histories when each query arrives and find ρ_t^c-neighbors dynamically. The main benefit of this method is that we can begin from a large ρ_t^c to seek more information, which can be utilized to overcome the cold start issue step by step. However, this dynamical partitioning method requires the storage and traversal of the complete query history, namely, $\mathcal{H}_t$, which increases the computation time and storage requirements. In many particular scenarios, service providers want to rent additional resources at lower cost over the cloud for computing and storage and accelerate the retrieval process; in such cases, the cold start issue may be less important.

To address this case, we propose a modified algorithm in this section that partitions the context space into several clusters for similar contexts before the first query arrives and builds a cover tree for each cluster to store variables such as the empirical average reward and retrieval times. When a query arrives, the algorithm finds the cluster to which c_t belongs and explores the corresponding cover tree to find the optimal video.

Algorithm 3 Static Adaptive Cover Tree Algorithm

```
1: Input : Parameters $\nu_1 > 0$, $\rho \in (0,1)$, $c > 0$, confidence $\delta \in (0,1)$, time horizon n and cover tree structure $(\mathcal{P}_{h,i})_{h\geq 0, 1\leq i\leq 2^h}$.
2: Partition context space into $m_c$ parts.
3: Initialize : $t = 1$; For all $a \in [1, m_c]$: $\mathcal{T}_t^a = \{(0,1),(1,1),(1,2)\}, H^a(t) = 1, U_{1,1}^a(t) = U_{1,2}^a(t) = C_{max}$.
4: loop
5:     if $t = t^+$ then
6:         for all $a \in [1, m_c]$ do
7:             Update tree $\mathcal{T}_t^a$, where the process is the same as in lines 5-8 in Alg. 1.
8:         end for
9:     end if
10:    The user inputs the current context $c_t$; find $b_{a_t}$.
11:    $(h_t, i_t), P_t \leftarrow Optpath(\mathcal{T}_t^{a_t})$.
12:    The user browses the video that corresponds to $v_{h_t,i_t}$ and gives his/her feedback.
13:    Go to the next time slot: $t = t + 1$.
14:    Update the browsing counter $T_{h_t,i_t}^{a_t}(t)$ and the empirical average reward $\widehat{\mu}_{h_t,i_t}^{a_t}(t)$.
15:    Update estimated retrieval result upper bound $U_{h_t,i_t}^{a_t}(t)$.
16:    UpdatePath($\mathcal{T}_t^{a_t}, (h_t, i_t), P_t$).
17:    Explore the new node via the same process as in lines 12-15 in Alg. 1.
18: end loop
```

Formally, we separate the context space into m_c parts, and each part is a subspace with diameter ρ_n^c (denoted as $b_1, b_2, ..., b_{m_c}$), where $m_c \leq C_C(\rho_n^c)^{-d_C}$ is the covering number of the context space and n is the time horizon, which is known in advance. Given two contexts $c, c' \in b_a$ ($a \in [1, m_c]$), for any video $v \in \mathcal{V}$, we have the following: $|\mu_{v,c} - \mu_{v,c'}| \leq L_C\rho_n^c$.

An illustration is shown in Fig. 3. As contexts arrive, SACT puts them into the corresponding regions. Each black point represents a historical record, which has been printed to facilitate understanding. However, for SACT, all records that are in the same subspace will be amalgamated into one. Comparing it with the ACT example in Fig. 1, SACT partitions the context space into the smallest part at the beginning, which causes the cold start issue. According to the scenario that is highlighted in the orange circle, the most valuable historical result may be stored in another subspace, rather than the part to which c_t belongs, which will cause a slight inaccuracy when comparing it with the ACT, the price of decreasing the time and space complexities.

For each context subspace b_a, we build a cover tree $\mathcal{T}_a$. Let $\mathcal{T}^s = \{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_{m_c}\}$ be the set of all cover trees. The notations $\mathcal{T}_t^a$ and $H^a(t)$ have the same meanings as $\mathcal{T}_t$ and $H(t)$, respectively, as defined in Section IV for tree $\mathcal{T}_a$, and a_t denotes the number of context subspaces to which c_t belongs. Since the algorithm partitions the context space in advance but not dynamically, we call it *Static Adaptive Confidence Tree*; it is presented as Alg. 3.

Now, we must redefine various notations. The set of arrival times of the contexts that have arrived in subspace b_a up to the current time t is denoted as

$$\mathcal{T}(b_a) = \{\tau : \tau < t, c_\tau \in b_a\}.$$

The number of times that node (h,i) in tree $\mathcal{T}^a$ has been browsed is expressed as

$$T_{h,i}^a(t) = \sum_{\tau \in \mathcal{T}(b_a)} \mathbb{I}\{h_\tau = h, i_\tau = i\}.$$

The empirical retrieval result $\widehat{\mu}_{h,i}^a(t)$ of the expected reward of $v_{h,i}$ if the context belongs to b_a is computed as

$$\widehat{\mu}_{h,i}^a(t) = \frac{1}{T_{h,i}^a(t)} \sum_{\tau \in \mathcal{T}(b_a)} r_\tau.$$

The other notations ($U_{h,i}^a(t)$, $B_{h,i}^a(t)$, and $H^a(t)$), which we do not define again but to which we add a superscript a, have the same meaning for the corresponding tree as in Section IV.

Since the relationship between each tree and context partition has been determined at the beginning, and the past selection time for each node is certain, the update function, namely, *UpdatePath*, does not need to update the whole tree every time; instead, it must update only the path from the root to the selected node (Alg. 4). Moreover, the third term of $U_{h,i}^a(t)$, which is the upper confidence bound, is changed when $t = t^+$; thus, we must refresh all trees with phase $\mathcal{O}(log(n))$. This is one of the greatest differences between SACT and ACT.

The two algorithms adopt the same exploration condition such that the conclusion of Lemma 1 is valid for SACT holds. Meanwhile, for SACT, Lemma 3 holds, which draws the same conclusion as Lemma 2 but considers a different choice of $\widetilde{\delta}(t)$ (see the detailed definition and proof in Appendix B).

Theorem 2 (Regret Bound of SACT). *Let users' feedback be i.i.d. and assumptions 1 and 2 hold at each time slot t. Then, the expectation regret of SACT R_n with time horizon n is*

$$\begin{aligned} R_n \leq\ & \frac{6\delta}{5} + 6L_c m + \frac{6\nu_1}{\rho} n^{\frac{d_V+d_C+1}{d_V+d_C+2}} (\log n)^{\frac{1}{d_V+d_C+2}} \\ & + \frac{6L_c(d_V + d_C + 2)}{d_V + d_C + 1} n^{\frac{d_V+d_C+1}{d_V+d_C+2}} (\log n)^{\frac{1}{d_V+d_C+2}} \\ & + \frac{96 C_V C_C \rho^{d_V - 1} \log(m_c \frac{\sqrt[8]{3\nu_1} n}{\delta \sqrt[8]{\rho}})}{(1 - \rho^{(1+d_V)})\nu_1 \nu_2{}^{d_V}(1-\rho)} (\frac{n}{\log n})^{\frac{d_V+d_C+1}{d_V+d_C+2}}. \end{aligned}$$

Proof: See the detailed proof in [59]. ■

According to the above result, SACT has sublinear regret $R_n = \mathcal{O}(n^{\frac{d_V+d_C+1}{d_V+d_C+2}} (\log n)^{\frac{1}{d_V+d_C+2}})$ as well; hence, both of our algorithms have the same convergence rate. To ensure that m_c is not too large, d_C must be fixed. Techniques for reducing the dimension may be useful (we used the method in [55] in our experiments).

The regret upper bound of SACT has the same order as that of ACT; however, SACT suffers from the cold start issue. To mitigate this issue, we can merge the static partition "dynamically". Specifically, the context partitioning and the tree building follow the same method as in SACT. When a query arrives, we combine the tree to which c_t belongs with its neighbors' to make decisions, e.g., calculate the average of $\widehat{\mu}_{h,i}(t)$ and sum $T_{h,i}(t)$ (for an unexplored node, set both to 0). All combined trees must be updated according to feedback. Then, the number of combined neighbors decreases gradually. This approach can mitigate the cold start issue by sacrificing running time and maintain the sublinear storage complexity.

VI. Complexity Analysis

In this section, we discuss the complexities of our algorithms and prove that SACT requires only sublinear storage. Then, we compare the regret upper bound and time and

TABLE II: Comparison with existing large-scale bandit algorithms

	Regret	Space complexity	Time complexity	Context	Infinite arms
ACR [51]	$\mathcal{O}(T^{\frac{d_I+d_C+1}{d_I+d_C+2}}\log T)$	$\mathcal{O}(\sum_{l=0}^{E} K_l + T)$	$\mathcal{O}(T^2+K_E T)$	Yes	No
HCT [57]	$\mathcal{O}(T^{\frac{d+1}{d+2}}(\log T)^{\frac{1}{d+2}})$	$\mathcal{O}(\log T^{\frac{2}{d+2}} T^{\frac{d}{d+2}})$	$\mathcal{O}(T\log T)$	No	Yes
ACT	$\mathcal{O}(T^{\frac{d_V+d_C+1}{d_V+d_C+2}}(\log T)^{\frac{1}{d_V+d_C+2}})$	$\mathcal{O}(T)$	$\mathcal{O}(T^2)$	Yes	Yes
SACT	$\mathcal{O}(T^{\frac{d_V+d_C+1}{d_V+d_C+2}}(\log T)^{\frac{1}{d_V+d_C+2}})$	$\mathcal{O}(\log T^{\frac{2}{d_V+d_C+2}} T^{\frac{d_V+d_C}{d_V+d_C+2}})$	$\mathcal{O}(T(T/\log T)^{\frac{d_C}{d_V+d_C+2}})$	Yes	yes

Algorithm 4 The *UpdatePath* function

```
1: Input : Tree $\mathcal{T}_t^{a_t}$, the path $P_t$, browsed node $(a_t, h_t, i_t)$.
2: if $(a_t, h_t, i_t) \in leaf(\mathcal{T}_t^{a_t})$ then
3:     $B_{h_t,i_t}^{a_t}(t) \leftarrow U_{h_t,i_t}^{a_t}(t)$.
4: else
5:     $B_{h_t,i_t}^{a_t}(t) \leftarrow \min[U_{h_t,i_t}^{a_t}(t), \max_{j\in\{2i-1,2i\}} B_{h_t+1,j_t}^{a_t}(t)]$.
6: end if
7: for all $(a_t, h, i) \in P_t - (a_t, h_t, i_t)$ backward do
8:     $B_{h,i}^{a_t}(t) \leftarrow \min[U_{h,i}^{a_t}(t), \max_{j\in\{2i-1,2i\}} B_{h+1,j}^{a_t}(t)]$.
9: end for
```

space complexities with those of existing large-scale bandit algorithms.

Space complexity. The following theorem bounds the space complexity of SACT.

Theorem 3. *Let $\mathcal{N}_T$ denote the space complexity of SACT up to time T. Under the same condition as in Theorem 2, we have*

$$\mathbb{E}(\mathcal{N}_T) = \mathcal{O}(\log T^{\frac{2}{d_V+d_C+2}} T^{\frac{d_V+d_C}{d_V+d_C+2}}).$$

Proof: See the detailed proof in [59]. ■

Theorem 3 proves that our SACT requires only sublinear storage: we just have to store the corresponding empirical rewards and visiting times of each node of the tree. However, for ACT, we must store all historical information (rewards, watched videos and contexts) and the tree structure separately. In the worst case, where only one kind of context arrives all the time and the tree has been fully explored, Theorem 3 is valid for ACT as well if $m_c = 1$. In practical scenarios, diverse users come with different contexts, which causes the number of nodes to be much smaller than this bound. Hence, the storage of ACT is mainly occupied by historical results $\mathcal{H}_T$, which is $\mathcal{O}(T)$. Although our approaches cannot reduce the video storage over the media cloud, due to their low storage complexity, they only slightly increase the storage burden on the cloud. Theorem 3 will be used to bound the time complexity.

Time complexity. The computational cost of ACT can be divided into three parts: finding ρ_t^c-neighbors, refreshing the cover tree and traversing the tree to retrieve the optimal video. For the first part, ACT must traverse all histories at each time t, for which the computational cost is $\mathcal{O}(t)$. Since Theorem 3 is valid for ACT as well, the number of video clusters to update cannot exceed $\mathcal{O}(\log T^{\frac{2}{d_V+d_C+2}} T^{\frac{d_V+d_C}{d_V+d_C+2}})$. Since the boundedness of the depth is at most $\mathcal{O}(\log t)$, the cost of traversing the tree is no more than $\mathcal{O}(\log t)$. Hence, up to time T, the computational complexity is $\mathcal{O}(T^2 + \log T^{\frac{2}{d_V+d_C+2}} T^{1+\frac{d_V+d_C}{d_V+d_C+2}} + T\log T) = \mathcal{O}(T^2)$.

For SACT, by Theorem 3, the number of video clusters that must be updated is at most $\mathcal{O}(\log T^{\frac{2}{d_V+d_C+2}} T^{\frac{d_V+d_C}{d_V+d_C+2}})$ with refresh phase $\mathcal{O}(\log t)$. Additionally, at each time t, SACT must find the context partition c_t from $\mathcal{O}((t/\log t)^{\frac{d_C}{d_V+d_C+2}})$ trees. Then, the cost of both traversing the tree and updating B and U is $\mathcal{O}(\log t)$. As a result, the total computational cost up to time horizon T is $\mathcal{O}((\log T/T)^{\frac{2}{d_V+d_C+2}} T\log T + T(T/\log T)^{\frac{d_C}{d_V+d_C+2}} + T\log T) = \mathcal{O}(T(T/\log T)^{\frac{d_C}{d_V+d_C+2}})$.

We list the regrets and complexities of our two algorithms and two existing large-scale bandit algorithms in Table II. SACT can greatly reduce the computational time and storage. Compared with [51], our algorithms have much lower regret and space complexity. Meanwhile, when the number of videos is not sufficiently large, the ACR in [51] costs less time than ACT. However, as the number of videos increases, our approach becomes faster. Hence, our algorithms are more suitable for big data video retrieval problems. Specifically, if $(\log T/T)^{\frac{2}{d_V+d_C+2}} T < K_E$, ACT has a smaller computational complexity, where K_E is the actual number of items.

VII. Numerical Results

In this section, we demonstrate the performance of theoretical regret bounds for our algorithms with experimental results that are based on large-scale real-world datasets. We evaluate our algorithms in terms of two aspects: 1) learning performance and 2) personalization. The main objective of the first aspect is to compare our algorithms with other online learning and hashing approaches to evaluate the learning performance as an online learning algorithm. However, owing to the serious logistical challenges of building a system in which to run the personalized algorithms on "live" data, all experiments are run on previously collected offline datasets, and we build a testbed, which is similar to that in [33], that consists of a video space, a set of queries, and a set of hypothetical context scenarios for evaluating the accuracy of personalized retrieval. Next, we evaluate the personalization with real-world video searching events, on which the hashing method cannot be implemented, unfortunately.

A. Dataset Description

In this work, we use one video dataset and one image dataset to evaluate the performance. For the video dataset, we crawled 121,460 videos with raw text features, including name, uploader, categories, duration, rating, views, number of comments, tags, synopsis and hot comments from YouKu[2]. Then, we aggregated all text information (videos name, views, tags, synopsis and hot comments) into a document and ran

[2]http://www.youku.com/
API: http://open.youku.com/

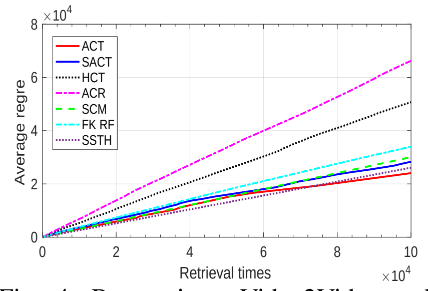

Fig. 4: Regret in a Video2Video task (static dataset).

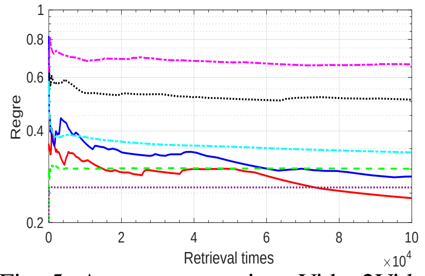

Fig. 5: Average regret in a Video2Video task (static dataset).

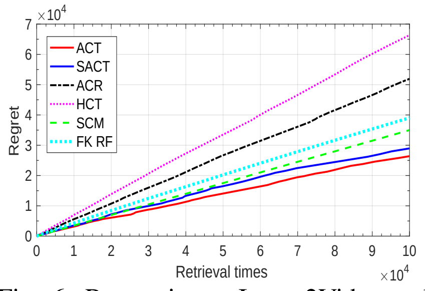

Fig. 6: Regret in an Image2Video task (static dataset).

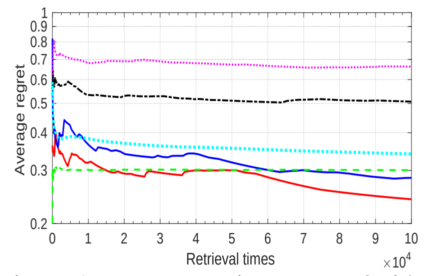

Fig. 7: Average regret in an Image2Video task (static dataset).

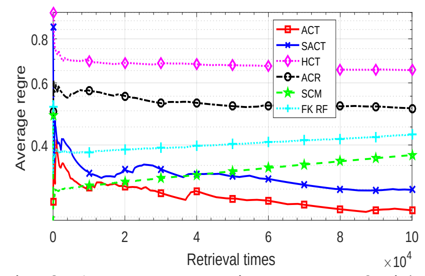

Fig. 8: Average regret in an Image2Video task (expanding dataset).

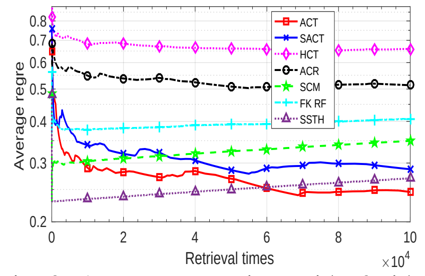

Fig. 9: Average regret in a Video2Video task (expanding dataset).

the latent Dirichlet allocation (LDA) algorithm to obtain a 10-dimensional topic vector. In sum, we obtain a 15-dimensional feature for describing a video. Since there is no need for our method to use similarity matching, we do not need to extract keyframes or HLFs, which substantially reduces our workload in preprocessing data.

For the personalized retrieval experiment, we randomly recorded 5.1 million searching events by 568,361 users in YouKu from Dec. 17-23, 2016. Each user's searching event consists of four components: 1) user information: ID, gender, location, birthday and personal profile; 2) the search time: only the number of hours is recorded; 3) the text input for query; and 4) the search result: the first ten results and whether users click them or not. Then, we deleted events that do not have sufficient user information and obtained 2.3 million events by 143,265 users.

For the image dataset, we adopted the public **NUS-WIDE** dataset. NUS-WIDE contains 269,648 images that have been downloaded from Flickr, with a total number of 5,018 unique tags. Ground-truth tags for 81 concepts can be utilized for evaluation. Each image is represented by a series of features: a 64-D color histogram, a 144-D color correlogram, a 73-D edge direction histogram, 128-D wavelet texture, 225-D block-wise color moments and a 500-D bag of words that is based on SIFT descriptors. We randomly select data from 100,000 images as queries.

For all aforementioned features, we adopt the methods that are described in [55] to reduce the dimensionality of the items. Finally, we obtain a 5-D feature vector of an image and a 9-D feature vector of a video. According to these final vectors, we set the parameter $\nu_1 = \sqrt{17}$ to guarantee that Assumption 1 is satisfied. When expanding the cover tree, we divide each cluster equally; thus, $\rho = 1/2$. To ensure the performance, we empirically set $\delta = 0.05$. For simplicity, we take the Euclidean norm as the dissimilarity function for both spaces.

B. Experimental Settings

1) Testing Learning Performance: The evaluation of personalization is known to be a difficult and expensive task. To demonstrate the improvement in personalization performance, we encoded the collected user information as 3-D vectors, reduced the searching time and aggregated them with a query as context input. To demonstrate the ability to handle queries of various types, we design two tasks for evaluating the learning performance: in *Image2Video*, we use an image in NUS-WIDE to retrieve video dataset ($d_C = 5 + 3, d_V = 9$), and in *Video2Video*, the query video is contained in video database ($d_C = 9 + 3, d_V = 9$).

To simulate users' feedback, we must select the ground truth. For the Image2Video task, we use all 81 concepts that are provided by NUS-WIDE to search in YouKu. The top 50 videos are labeled as the ground truth, and we define an event of successful retrieval, which is denoted as a_t, as a video v_t that is labeled by the query image's tags. For the Video2Video task, we use video's title to search in YouKu, and the top 30 videos are labeled as the relevant videos. A successful retrieval event, which is denoted as a_t, is defined as a relevant video v_t of the query. Then, the precision (reward) and the regret at time t can be computed as $r_t = s_{v_t}\mathbb{I}_{a_t}/10$ and $\Delta_t = 1 - r_t$, where $s_v \in [0, 10]$ is based on the score of video/query/user tuples that we rated manually.

2) Personalized Retrieval: For the personalized retrieval experiment, we demonstrate how much the proposed personalized retrieval algorithm can influence the results and compare our work with others over a real-world searching dataset. Under the experimental datasets that are specified above, we run the five algorithms *with context* (contextual) and *without context* (context-free) and set $T = 100000$. For the contextual task, users' information, searching time and text input are encoded and reduced to $d_C = 5$ as a contextual group via the process that is discussed above. Then, we only encode the searching time and text input as a context-free group. To remove the effect of the number of context dimensions, it is generated as $d_C = 5$ by setting all users' information to the same values. Two groups will be used as input to retrieve the video dataset ($d_V = 5$) separately.

To obtain an unbiased evaluation of our online algorithms on this offline dataset, we run an experiment that is similar to that in [55]: at each time slot, we repeatedly and randomly select a searching event until the output v_t belongs to the corresponding searching result. Then, we set the precision (reward): $r_t = \mathbb{I}\{v_t \text{ was clicked}\}$.

3) Methods to be Compared: To evaluate the retrieval quality, we compare the proposed algorithms with several state-of-the-art algorithms: two large-scale bandit algorithms, namely, the adaptive clustering recommendation algorithm (ACR) [51] and the high-confidence tree algorithm (HCT) [57]; a cross-modal supervised hashing algorithm, namely, semantic correlation maximization (SCM) [60]; and a relevance feedback algorithm, namely, the Fisher kernel (FK) relevance feedback approach [43]. For the Video2Video task, we also compare our algorithm with a state-of-the-art content-based hashing algorithm: SSVH [24]. Most codes and suggested parameters of these methods are available from the corresponding authors, and we set the code length to 32 bits. HCT is a context-free algorithm, as it views all inputs as being of the same type. To modify it into a retrieval problem, we simply aggregate the d_C^q-dimensional query conditions and the d_V-dimensional video vectors as the arm of HCT. For hashing methods, we take 5% of datasets as the query sets and the rest as training sets and the retrieval database. For ACR, the epoch $E = min\{L, \lfloor \log_2(T) \rfloor\}$. Since each leaf node in ACR only contains one item, the depth of the tree is $L = 17$, and the time horizon $T = 10^5$. Thus, $E = \lfloor \log_2(T) \rfloor = 16$. For the FK RF algorithm, we select their Global FK RBF framework since frame aggregation FK RBF is too slow, as it requires more than 4 seconds for each retrieval.

In addition, we report the time and space consumption of all algorithms. All the experiments are implemented and run on our university's high-performance computing platform, whose GPU reaches 18.46 TFlops and SSD cache is 1.25 TB.

C. Result on Testing Learning Performance

In this subsection, we show the following: 1) our regret bounds are sublinear and the time-averaged regret converges to 0; 2) our algorithms can handle various query types; 3) our algorithms do not suffer severely from the cold start problem; 4) SACT can substantially decrease the time and storage complexities; and 5) our approaches are scalable.

1) Overall Comparison with Baseline: First, we run the experiment on a static dataset, which retrieves the whole video dataset at the beginning. We present the results of the Video2Video task in Fig. 4 and Fig. 5 and the Image2Video task in Fig. 6 and Fig. 7. From the above four figures, we observe the following: 1) compared with other learning methods, our algorithms have lower regret bounds and converge much faster in the large-scale setting; 2) compared with other retrieval methods, our proposed online learning algorithms can learn while continuously improving the retrieval accuracy; 3) the hashing method performs worse than the proposed contextual online learning method, as it has a poor ability to process the stochastic result of personalized retrieval; 4) based on the nearest-neighbor search, the FK RF method has a smoother average regret decline, but its poor performance renders it unsuitable for personalized video big data retrieval; and 5) the regret for the Video2Video task is much better compared to the Image2Video task, especially for SCM. We use a higher dimension requirement for the query, which means that users provide more detailed information. Moreover, using videos as queries strengthens the connection between the query and the retrieved videos. However, the selection of the ground truths for various tasks may also impact the regret. In the Video2Video task, the score of the task depends more on similarity compared to the Image2Video task. For the same reason, the regret of the Video2Video task changes more smoothly than the regret of the Image2Video task.

Furthermore, in Fig. 5 and Fig. 7, fluctuations are observed at the beginning of the learning method. However, our algorithms learn the map from queries to videos fast and enter a smooth descent stage of average regret, which demonstrates overcoming of the cold start issue. Compared with the ACT, SACT suffers a longer and more severe fluctuation period, which is in line with our expectations.

Next, to demonstrate the ability to handle big data, we use an expanding dataset. We generate the expanding dataset by setting 20,000 videos as the initial video space. Then, the rest of the videos are added to the video space randomly at each round. The comparison results of regret are presented in Fig. 8 and Fig. 9. Our proposed online learning methods have inconspicuous and acceptable performance loss when facing the expanding dataset. The long-term average regrets of ACT and SACT are not influenced substantially. However, for similarity-based methods, SCM, SSVH, and FK RF, the loss increases nearly linearly. Hence, our top-down tree structure can support big data problems.

In addition, we present the results of the Image2Video task with various query dimensions in Fig. 10. Specifically, we run our ACT algorithm with three context dimensions: the 3-D feature that is described above, the 64-D color histogram features that are provided by the dataset, and only the 1-D concept feature, which contains 81 ground-truth concepts. From Fig. 10, the total regret up to time 10^5 with the 5-D feature is 69.42% lower than that with the 64-D feature and 71.81% lower than that with the 1-D feature. This result demonstrates that query dimensions that are too high/low may cause redundancy/lack of information and lead to a large regret; meanwhile, different features may lead to different retrieval performances.

2) Space Cost: In this subsection, we study the space costs of the Image2Video task for our two algorithms compared with other learning methods. Space costs of up to 100,000 retrievals are recorded and listed in Table III. This storage does not include the space cost that is incurred by the video dataset.

According to the table, our two algorithms have acceptable space costs. SACT dramatically reduces the storage; hence,

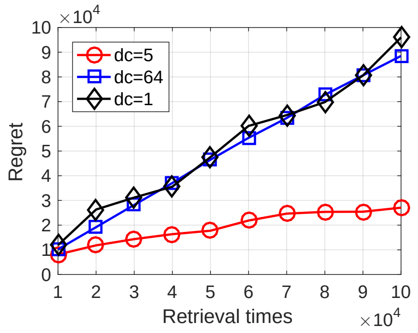

Fig. 10: Regret in Image2Video tasks with various query dimensions.

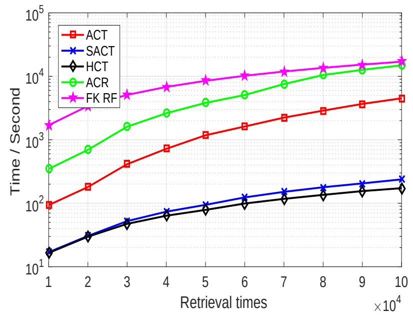

Fig. 11: Computational cost of Image2Video task ($d_c = 8$).

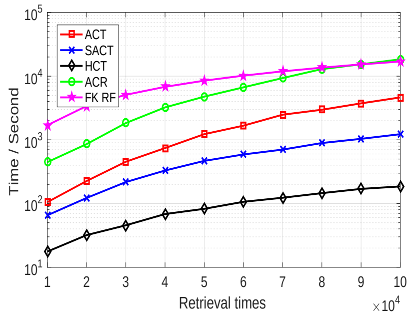

Fig. 12: Computational cost of Image2Video tasks ($d_c = 12$).

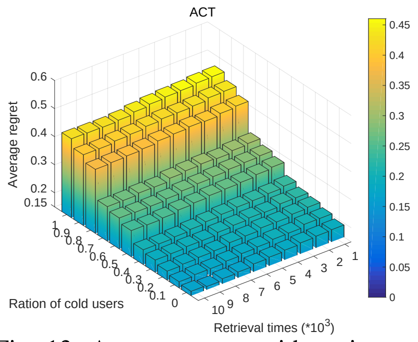

Fig. 13: Average regret with various ratios of cold users.

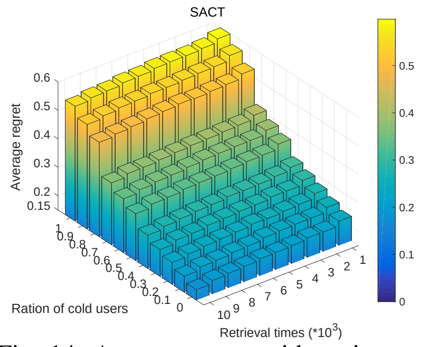

Fig. 14: Average regret with various ratios of cold users.

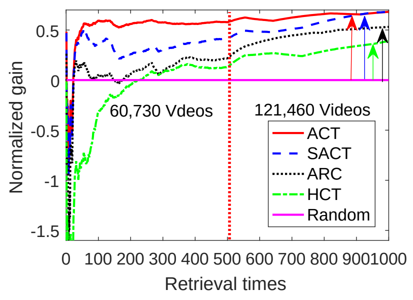

Fig. 15: Scalable performances of all algorithms.

TABLE III: Space cost (kb)

Algorithm	Retrieval Times $\times 10^4$				
	0.1	1	5	8	10
ACT	46.1	461.6	2123.4	3379.6	4253.9
SACT	32.8	221.8	901.4	1158.3	1469.1
HCT	48.7	322.7	1168.9	1609.2	1920.5
ACR	102.6	967.2	4542.0	7096.4	8799.7

it is highly suitable for big data problems. Another important finding is that the impact of increasing the dimensions is more serious for SACT than ACT because the increasing dimensions cause ACT to move far away from the worst case, in which contexts of only one type are arriving.

3) Time Cost: We investigate the running times of the proposed ACT and SACT algorithms in comparison with the baselines of bandit methods. We conduct experiments on two tasks to analyze the time consumption. The time costs of each method are shown in Fig. 11 and Fig. 12. The following are observed. 1) The average running times of ACT for a single retrieval are $0.0464/0.04501s$ for the two tasks, and those of SACT are $0.01228/0.002376s$. A substantial amount of time is spent updating cover trees. By limiting the learning period, our proposed algorithms can realize a higher response speed. 2) SACT can substantially reduces the computational cost as well; however, it is highly influenced by d_C. In contrast, ACT still has a stable result. The results coincide with those for the space cost. Hence, these two algorithms are applicable to different scenarios. 3) ACR requires a substantial amount of time to traverse nodes. Thus, it cannot be applied to real large-scale scenarios. 4) HCT has the highest speed since it does not process the context data. Hence, HCT has low retrieval precision. 5) FK RF has a linear time cost at nearly 1.7 seconds preretrieval. In summary, the proposed algorithms can perform high-precision retrieval efficiently in big data problems.

D. Results on Personalized Retrieval

In this subsection, we present the results of utilizing context information. First, we define the learning gain of algorithm π_1 over algorithm π_2 as $(A_{\pi_1,t} - A_{\pi_2,t})/A_{\pi_2,t}$, where $A_{\pi,t}$ denotes the average accuracy of algorithm π at time t. Then, we present the average accuracies and gain in Table IV. Moreover, we calculate the gain of each algorithm in the contextual group over itself in the context-free group and list the results in Table V. ACT and SACT significantly outperform the existing works, especially the context-free algorithm. The RF method outperforms other online learning methods; however, it has weaker performance than our method, which demonstrates that the stochastic process feedback model performs better than the previous model. Comparing with the result in Fig. 4-7, there are notable improvements in accuracy for every algorithm. In our analysis, this improvement is because the score in the above experiments, which we rated manually, has higher randomness. However, in these real-world searching data, there are hot videos that the majority of people love, which result in a much easier learning process. Moreover, with the help of real-world users' context, SACT can converge to a similar performance to ACT after a long-term run. This result demonstrates that the sacrifice of SACT performance mainly concerns the issue of cold start and users' context can substantially alleviate the loss.

In Table V, we compare each algorithm with itself under two scenarios. Three contextual algorithms have an increase of approximately $6-10\%$ in average accuracy, whereas the

TABLE IV: Influence of context.

Task	Algorithm	Retrieval Times $\times 10^4$ (context-free)					Retrieval Times $\times 10^4$ (contextual)				
		0.1	1	5	8	10	0.1	1	5	8	10
Average Accuracies	ACT	63.23%	74.59%	81.58%	84.31%	84.91%	70.58%	80.63%	89.38%	90.20%	90.45%
	SACT	64.38%	73.82%	80.38%	82.63%	82.91%	69.21%	78.39%	88.64%	89.32%	89.35%
	HCT	35.31%	38.84%	39.88%	40.21%	40.38%	33.81%	38.65%	39.98%	39.33%	40.26%
	ACR	43.31%	49.81%	53.58%	55.23%	55.66%	47.46%	51.38%	58.81%	60.32%	61.16%
	FK RF	54.62%	57.38%	61.42%	64.38%	65.02%	55.12%	57.64%	61.43%	63.96%	65.14%
Gain	ACT over HCT	79.07%	92.04%	104.56%	110.97%	110.28%	108.75%	108.62%	123.56%	129.34%	124.14%
	SACT over HCT	82.33%	90.06%	101.55%	105.50%	105.32%	104.70%	102.82%	121.71%	127.10%	121.93%
	ACT over ACR	45.99%	49.45%	52.26%	53.59%	52.55%	48.71%	56.93%	51.98%	49.53%	47.55%
	SACT over ACR	48.65%	48.20%	50.02%	49.61%	48.96%	45.83%	52.57%	50.72%	48.08%	46.09%
	ACT over FK RF	15.76%	30.00%	32.82%	31.76%	30.59%	28.05%	39.89%	45.50%	41.03%	38.53%
	SACT over FK RF	17.87%	28.65%	30.87%	28.35%	27.51%	25.56%	36.00%	44.29%	39.65%	37.17%

TABLE V: Gain of the contextual group over the context-free group.

	Retrieval Times $\times 10^4$				
	0.1	1	5	8	10
ACT over ACT	11.62%	8.10%	9.56%	6.33%	6.28%
SACT over SACT	7.50%	6.19%	10.28%	8.10%	7.77%
HCT over HCT	–4.25%	-0.49%	0.25%	-2.20%	-0.30%
ACR over ACR	9.58%	3.15%	9.76%	9.22%	9.88%
FK FR over FK FR	0.92%	0.45%	0.02%	-0.65%	0.18%

performance of HCT changes little. ACT can obtain more information than SACT at the beginning; hence, the increase in the first 1000 retrievals is higher, which can be up to 11.62%. More importantly, the largest increase that is obtained by using users' conditions is not in the retrieval accuracy but in the learning speed of the algorithms. The algorithms that use context can obtain much higher accuracy in less time. Therefore, we demonstrate from another aspect that our contextual algorithms overcome the cold start issue.

1) Cold Start Issue: To further demonstrate the ability to deal with new users, we randomly select $20,00$ users with various ratios of *cold users* to compose 11 new datasets (e.g., $0\%, 10\%, and 20\% \cdots$), where a *cold user* is defined as a user who has performed fewer than 5 searches. We repeat the experiment 10 times on each dataset, record the average results, and show them in Fig. 13 and Fig. 14. According to the two figures, the influence of cold users can be divided into three cases. When the ratio is less than (or equal to) 50%, the ratio of cold users has a small effect, and our algorithms have similar accuracy in these datasets. When the ratio is between 50% and 80%, the regret increases slightly. When the ratio exceeds 80%, the regret increases substantially because the datasets are full of cold users, and it is difficult to find useful histories. However, even in the worst-case scenario, the average regrets decrease over time, especially for ACT. Hence, the problem still can be learned, although a longer time is required. As a result, our approaches can deal with the cold start issue. Comparing the results of the two methods, SACT suffers from the cold start issue slightly more severely. Thus, although ACT can continuously improve the accuracy, SACT performs better in terms of storage and running time, which will be demonstrated later.

2) Scalability: In this subsection, we demonstrate that our algorithms have the ability to scale up. In particular, we divide the video dataset into two parts, of which each has half the videos of the complete dataset. First, the algorithms are only run on one part of the dataset. When $t = 500$, we add another part into the experiment to simulate scaling up of videos.We calculate the gains of all algorithms over the random algorithm and show the result in Fig. 15. After $t = 500$, none of the four algorithms has substantial performance loss since all of them are scalable. The performance increases after $t = 500$ are due to the decreased performance of the random algorithm. The performance increases against the random algorithm are 68.32%, 68.32%, 53.09%, and 39.04% for the ACT, SACT, ACR and HCT algorithms, respectively. Our algorithms outperform the others quickly since 500 retrieval times is a relatively short duration.

E. Sensitivity Analysis

In this subsection, we test the ability of the proposed algorithms in dealing with errors in the input. We add random noise to the user's context and evaluate the performances of two algorithms. In detail, we set $c'_t = c_t + 0.3p$, where p is a d_C-dimensional vector and each component obeys an i.i.d. Gaussian distribution $p^i \sim \mathcal{N}(0,1)$ $(i \in [1, d_C])$. We multiply p by a constant, namely, 0.3, to restrict the noise size and prevent the signal from being overwhelmed by the noise. The results of the learning performance testing are shown in Fig. 16 and Fig. 17.

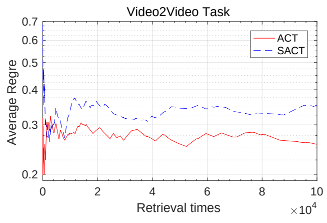

Fig. 16: Sensitivity analysis for the Video2Video task.

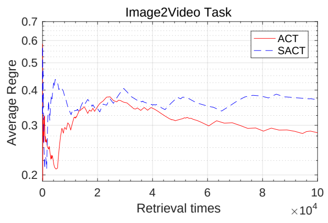

Fig. 17: Sensitivity analysis for the Image2Video task.

The noise has a strong influence on the algorithm performance, which causes irregular fluctuations of the average regret. Compared with the results in Fig. 5 and Fig. 7, in the Video2Video task the performance loss of ACT is 6.28%, the loss of SACT is 23.75%, and in the Image2Video task the performance losses of ACT and SACT are 6.97% and 27.80%, respectively. Our algorithms can learn the retrieval problem and achieve accurate results. ACT substantially outperforms

SACT, which demonstrates that the dynamic partitioning method yields a more robust input context.

Moreover, we evaluate the robustness of the two algorithms in the personalized retrieval scenario. We run the experiment 10 times and list the accuracy losses for two algorithms in Table VI.

From the results, the influence of noise on contextual retrieval is larger compared to context-free retrieval since for contextual retrieval our algorithms must learn more information from the input. Additionally, the influence on SACT is greater than on ACT. We also observe that the regret loss is lower at $t = 1000$ than at $t = 10^4$; when $t \geq 10^5$, the loss declines to an acceptable value (approximately 5.5% for ACT and approximately 20% for SACT). We conjecture that the reason that the loss initially rises and subsequently falls is that at $t = 1000$, our algorithms are still in the initial learning phase and have not built an accurate map from the input to the videos, whereas at $t = 10^4$, without noise, our algorithms have learned the retrieval problem relatively accurately; however, noise obstructs this process severely. Nevertheless, the results demonstrate that over a long run, our contextual online learning approach can still learn the personalized retrieval problem.

TABLE VI: Influence of noise.

Task	**Algorithm**	**Retrieval Times** $\times 10^4$			
		0.1	1	10	100
Contextual	ACT	16.32%	37.43%	6.72%	5.63%
	SACT	12.59%	46.67%	26.04%	20.93%
Context-free	ACT	15.44%	35.75%	5.93%	5.17%
	SACT	16.38%	46.81%	24.60%	18.23%

VIII. Conclusions

In this paper, we propose two contextual online learning algorithms, namely, ACT and SACT, for the large-scale video retrieval problem. We use the contextual bandit framework and prove a sublinear regret bound. The main benefits of our approaches are as follows: 1) they fully utilize the users' feedback to achieve better results; 2) as they learn, the average error converges to zero; and 3) they do not require similarity matching and have more general applications in big data settings. To relate a user's query to a video, we use the concept of context and consider users' real-time conditions. In future works, determining how to further reduce the time and space complexities may be a fundamental objective of our investigations.

References

[1] C. G. M. Snoek, B. Huurnink, L. Hollink, M. de Rijke, G. Schreiber, and M. Worring, "Adding semantics to detectors for video retrieval," *IEEE Trans. Multimedia*, vol. 9, no. 5, pp. 975-985, Aug., 2007.

[2] B. V. Patel and B. B. Meshram, "Content Based Video Retrieval Systems," *Int. J. UbiComp* (IJU), vol. 3, no. 2, Apr., 2012.

[3] J. Davidson, B. Liebald, J. Liu, P. Nandy, and T. Van Vleet, "The YouTube Video Recommendation System," in *Proc. Fourth ACM Conf. Recommender Systems*, pp. 293-296, 2010.

[4] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra, "Relevance Feedback: A Power Tool for Interactive Content-Based Image Retrieval," *IEEE Trans. Circuits Video Technol.*, vol. 8, no. 5, pp. 644-655, 1998.

[5] p. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time abalysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, pp. 235-256, 2002.

[6] J. Song, Y. Yang, Z. Huang, H. Shen, and J. Luo, "Effective multiple feature hashing for large-scale near-duplicate video retrieval," *IEEE Trans. Multimedia*, vol. 15, no. 8, pp. 1997-2008, Dec., 2013.

[7] A. Araujo, J. Chaves, R. Angst, and B. Girod, "Temporal aggregation for large-scale query-by-image video retrieval," in *Proc. 22nd IEEE Int. Conf. Image Process.*, pp. 1519-1522, 2015.

[8] Y. Gu, C. Ma, and J. Yang, "Supervised Recurrent Hashing for Large Scale Video Retrieval," in *Proc. ACM Int. Conf. Multimedia*, pp. 272-276, 2016.

[9] S. Bubeck, N. Cesa-Bianchi, "Regret analysis of stochastic and non-stochastic multi-armed bandit problems," *Foundations and Trends in Machine Learning*, no. 5, vol. 1, pp.1-122, Dec. 2012.

[10] W. X. Zhao, S. Li, Y. He, E. Y. Chang, J. R. Wen, and X. Li, "Connecting Social Media to E-Commerce: Cold-Start Product Recommendation Using Microblogging Information," *IEEE Trans. Knowledge and Data Engineering*, vol. 28, no. 5 pp. 1147-1159, May, 2016.

[11] W. Zhu, C. Luo, J. F. Wang, and S. P. Li, "Multimedia cloud computing," IEEE *Signal Process. Mag.*, vol. 28, no. 3, pp. 59-69, 2011.

[12] W. Hu, N. Xie, L. Li, X. Zeng, and S. Maybank. "A Survey on Visual Content-Based Video Indexing and Retrieval," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 41, no. 6, 797-819, Nov. 2011.

[13] H. S. Chang, S. S. Sull, and S. U. Lee, "Efficient video indexing scheme for content-based retrieval," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 8, pp. 1269-1279, Dec. 1999.

[14] D. P. Mukherjee, S. K. Das, and S. Saha, "Key frame estimation in video using randomness measure of feature point pattern," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 5, pp. 612-620, May 2007.

[15] J. Meng, J. Yuan, J. Yang, G. Wang, and Y. P. Tan, "Object Instance Search in Videos via Spatio-Temporal Trajectory Discovery," *IEEE Trans. Multimedia*, vol. 18, no. 1, pp. 116-127, Jan., 2016.

[16] A. Anjulan and N.Canagarajah, "Aunified framework for object retrieval and mining," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 1, pp. 63-76, Jan. 2009.

[17] F. I. Bashir, A. A. Khokhar, and D. Schonfeld, "Real-time motion trajectory-based indexing and retrieval of video sequences," *IEEE Trans. Multimedia*, vol. 9, no. 1, pp. 58-65, Jan., 2007.

[18] C.-W. Su, H.-Y. M. Liao, H.-R. Tyan, C.-W. Lin, D.-Y. Chen, and K.-C. Fan, "Motion flow-based video retrieval," *IEEE Trans. Multimedia*, vol. 9, no. 6, pp. 1193-1201, Oct. 2007.

[19] J. Wang, T. Zhang, J. Song, N. Sebe, and H. Tao Shen, "A Survey on Learning to Hash," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 769-790, 2018.

[20] G. Ye, D. Liu, J. Wang, and S.-F. Chang, "Large-scale video hashing via structure learning," in *Proc. IEEE Int. Conf. Computer Vision*, pp. 2272-2279, 2013.

[21] S. Zhang, J. Li, J. Guo, and B. Zhang, "Scalable Discrete Supervised Multimedia Hash Learning with Clustering," *IEEE Trans. Circuits Syst. Video Technol.*, 2017.

[22] Y.-G. Jiang, Z. Wu, J. Wang, X. Xue, and S.-F. Chang, "Exploiting Feature and Class Relationships in Video Categorization with Regularized Deep Neural Networks," arXiv:1502.07209, 2015.

[23] H. Zhang, M. Wang, R. Hong, and T. Chua, "Play and Rewind: Optimizing Binary Representations of Videos by Self-Supervised Temporal Hashing," *Proc. ACM Int. Conf. Multimedia*, pp. 781-790, 2016.

[24] J. Song, H. Zhang, X. Li, L. Gao, M. Wang, and R. Hong, "Self-Supervised Video Hashing With Hierarchical Binary Auto-Encoder," *IEEE Trans. Image Process.*, vol. 27, no. 7, pp. 3210-3221, 2018.

[25] J. Song, L. Gao, L. Liu, X. Zhu, and N. Sebe, "Quantization-based hashing: a general framework for scalable image and video retrieval," *Pattern Recognition*, vol. 75, pp. 175-187, 2018.

[26] L. Shang, L. Yang, F. Wang, K.-P. Chan, and X.-S. Hua, "Real-time large scale near-duplicate web video retrieval," in *Proc. ACM Int. Conf. Multimedia*, pp. 531-540, 2010.

[27] H.-S. Min, J.-Y. Choi, W. De Neve, and Y.-M. Ro, "Nearduplicate video clip detection using model-free semantic concept detection and adaptive semantic distance measurement," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, pp. 1174-1187, 2012.

[28] X. Xu, F. Shen, Y. Yang, H. T. Shen, and X. Li. "Learning discriminative binary codes for large-scale cross-modal retrieval. *IEEE Trans. Image Processing*, vol. 26, no. 5, pp. 2494-2507, 2017.

[29] W. M. Hu, D. Xie, Z. Y. Fu, W. R. Zeng, and S. Maybank, "Semanticbased surveillance video retrieval," *IEEE Trans. Image Process.*, vol. 16, no. 4, pp. 1168-1181, Apr., 2007.

[30] S. Benini, L. Canini, and R. Leonardi, "A connotative space for supporting movie affective recommendation," *IEEE Trans. Multimedia*, vol. 13, no. 6, pp. 1356-1370, Dec., 2011.

[31] L. Canini, S. Benini, and R. Leonardi, "Affective Recommendation of Movies Based on Selected Connotative Features," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 4, pp. 636-647, 2013.

[32] H. Ghosh, P. Poornachander, A. Mallik, and S. Chaudhury, "Learning ontology for personalized video retrieval," in *Proc. ACM Workshop Multimedia Inform., Retrieval*, Augsburg, Germany, pp. 39-46, Sep., 2007.

[33] D.Vallet, P. Castells,M. Fernandez, P. Mylonas, and Y. Avrithis, "Personalized content retrieval in context using ontological knowledge," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 3, pp. 336-345, 2007.

[34] D. Tao, X. Tang, X. Li, and Y. Rui, "Direct Kernel Biased Discriminant Analysis: A New Content-Based Image Retrieval Relevance Feedback Algorithm," *IEEE Trans. Multimedia*, vol. 8, no. 4, pp. 716-727, 2006.

[35] X. S. Zhou, and T. S. Huang, "Relevance feedback in image retrieval: A comprehensive review," *Multimedia Syst.*, vol. 8, no. 6, pp. 536-544, 2003.

[36] N. Doulamis, and A. Doulamis, "Evaluation of relevance feedback schemes in content-based in retrieval systems," *Signal Process.: Image Commun.*, vol. 21, no. 4, pp. 334-357, 2006.

[37] N. Doulamis, and A. Doulamis, "Generalized nonlinear relevance feedback for interactive content-based retrieval and organization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 5, pp. 656-671, 2004.

[38] B. Demir, and L. Bruzzone, "A Novel Active Learning Method in Relevance Feedback for Content-Based Remote Sensing Image Retrieval," *IEEE Trans. Geosci. Remote Sens.*, vol. 53, no. 5, pp. 2323-2334, 2014.

[39] B. Leng, J. Zeng, M. Yao, and Z. Xiong "3D Object Retrieval With Multitopic Model Combining Relevance Feedback and LDA Model Sign In or Purchase," *IEEE Trans. Image Processing*, vol. 24, no. 1, pp. 94-105, 2014.

[40] X. Qian, X.g Tan, Y. Zhang, R. Hong, and M. Wang, "Enhancing Sketch-Based Image Retrieval by Re-Ranking and Relevance Feedback," *IEEE Trans. Image Processing*, vol. 25, no. 1, pp. 195-208, 2016.

[41] R. Yan, A. G. Hauptmann, and R. Jin, "Negative Pseudo-Relevance Feedback in Content-based Video Retrieval," *Proc. ACM Int. Conf. Multimedia*, pp. 343-346, 2003.

[42] L. Shao, S. Jones, X. Li, "Efficient search and localization of human actions in video databases," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 3, pp. 504-512, 2014.

[43] I. Mironică, B. Ionescu, J. Uijlings, and N. Sebe, "Fisher Kernel Temporal Variation-based Relevance Feedback for Video Retrieval," *Computer Vision and Image Understanding*, vol. 143, pp. 38-51, 2016.

[44] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades, "Facing the cold start problem in recommender systems," *Expert Systems with Applications*, vol. 41, pp. 2065-2073, 2014.

[45] G. Guo, J. Zhang, and D. Thalmann, "Merging trust in collaborative filtering to alleviate data sparsity and cold start," *Knowledge-Based Systems*, vol. 57, pp. 57-68, 2014.

[46] J. Lin, K. Sugiyama1, M. Kan, and T. Chua, "Addressing Cold-Start in App Recommendation: Latent User Models Constructed from Twitter Followers," *Proc. Int. ACM SIGIR nf. Res. Develop. Inf. Retrieval*, pp. 283-292, 2013.

[47] J. Tang, X. Zhang, and X. Xue, "Addressing Cold Start in Recommender Systems: A Semi-supervised Co-training Algorithm," *Proc. Int. ACM SIGIR nf. Res. Develop. Inf. Retrieval*, pp.73-82, 2014.

[48] M. Saveski, and A. Mantrach, "Item Cold-Start Recommendations: Learning Local Collective Embeddings," *Proc. 8th ACM Conf. Recommender Syst.*, pp. 89-96, 2014.

[49] I. Barjasteh, R. Forsati, D. Ross, A. Esfahanian, and Hayder Radha, "Cold-Start Recommendation with Provable Guarantees: A Decoupled Approach," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1462-1474, 2016.

[50] A. Levi, O. Mokryn, C. Diot, N. Taft, "Finding a Needle in a Haystack of Reviews: Cold Start Context-Based Hotel Recommender System," *Proc. 6th ACM Conf. Recommender Syst.*, pp. 115-122, 2012.

[51] L. Song, C. Tekin, and M. van der Schaar, "Online Learing in Large-scale Contextual Recommender System," *IEEE Trans. Services Computing*, vol. 9, no. 3, pp. 433-445, May-June, 2016.

[52] C. Tekin and M. van der Schaar, "Active learning in context-driven stream mining with an application to image mining," *IEEE Trans. Image Processing*, vol. 24, no. 11, pp. 3666-3679, 2015.

[53] J. Xu, D. Deng, U. Demiryurek, C. Shahabi, and M. van der Schaar, "Mining the situation: Spatiotemporal traffic prediction with big data," *J. Sel. Topics Signal Processing*, vol. 9, no. 4, pp. 702-715, 2015.

[54] D. Bouneffouf, A. Bouzeghoub, A. L. Gançarski, "A Contextual-Bandit Algorithm for Mobile Context-Aware Recommender System," in *Neural Information Processing*, Springer, pp. 324-331, 2012.

[55] L. Li, W. Chu, J. Langford, and R. E. Schapire. "A contextual-bandit approach to personalized news article recommendation," in *Proc. 19th Int. Conf. World Wide Web* (WWW 2010), ACM, pp. 661-670, 2010.

[56] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári, "X-Armed Bandits," *J. Mach. Learn. Res*, vol. 12, pp. 1655-1695, 2011.

[57] M. G. Azar, A. Lazaric, E. Brunskill, "Online Stochastic Optimization under Correlated Bandit Feedback," in *Proc. Int. Conf. Mach. Learn.* (ICML), Beijing, pp. 1557-1565, 2014.

[58] C. Tekin and M. van der Schaar, "Distributed online Big Data classification using context information," in *Proc. Int. Conf. Commun., Control, Comput.*, Oct., 2013.

[59] Y. Feng, P. Zhou, J. Xu, S. Ji and D. Wu, "Supplementary: Video Big Data Retrieval Over Media Cloud: A Context-aware Online Learning Approach". [Online] Available: https://www.dropbox.com/s/6oxbqtccblnr49b/supptmm.pdf?dl=0

[60] D. Zhang and W.-J. Li, "Large-scale supervised multimodal hashing with semantic correlation maximization, in *Proc. AAAI Conf. Artif. Intell.*, 2014, pp. 2177-2183.