Contextual Online Learning for Multimedia Content Aggregation

Cem Tekin, *Member, IEEE*, Mihaela van der Schaar, *Fellow, IEEE*

***Abstract*—The last decade has witnessed a tremendous growth in the volume as well as the diversity of multimedia content generated by a multitude of sources (news agencies, social media, etc.). Faced with a variety of content choices, consumers are exhibiting diverse preferences for content; their preferences often depend on the context in which they consume content as well as various exogenous events. To satisfy the consumers' demand for such diverse content, multimedia content aggregators (CAs) have emerged which gather content from numerous multimedia sources. A key challenge for such systems is to accurately predict what type of content each of its consumers prefers in a certain context, and adapt these predictions to the evolving consumers' preferences, contexts and content characteristics. We propose a novel, distributed, online multimedia content aggregation framework, which gathers content generated by multiple heterogeneous producers to fulfill its consumers' demand for content. Since both the multimedia content characteristics and the consumers' preferences and contexts are unknown, the optimal content aggregation strategy is unknown a priori. Our proposed content aggregation algorithm is able to learn online what content to gather and how to match content and users by exploiting similarities between consumer types. We prove bounds for our proposed learning algorithms that guarantee both the accuracy of the predictions as well as the learning speed. Importantly, our algorithms operate efficiently even when feedback from consumers is missing or content and preferences evolve over time. Illustrative results highlight the merits of the proposed content aggregation system in a variety of settings.**

***Index Terms*—Social multimedia, distributed online learning, content aggregation, multi-armed bandits.**

I. Introduction

A plethora of multimedia applications (web-based TV [1], [2], personalized video retrieval [3], personalized news aggregation [4], etc.) are emerging which require matching multimedia content generated by distributed sources with consumers exhibiting different interests. The matching is often performed by CAs (e.g., Dailymotion, Metacafe [5]) that are responsible for mining the content of numerous multimedia sources in search of finding content which is interesting for the users. Both the characteristics of the content and preference of the consumers are evolving over time. An example of the system with users, CAs and multimedia sources is given in Fig. 1.

Each user is characterized by its context, which is a real-valued vector, that provides information about the users' content preferences. We assume a model where users arrive sequentially to a CA, and based on the type (context) of the user, the CA requests content from either one of the multimedia sources that it is connected to or from another CA that it is connected to. The context can represent information such as age, gender, search query, previously consumed content, etc. It may also represent the type of the device that the user is using [6] (e.g., PDA, PC, mobile phone). The CA's role is to match its user with the most suitable content, which can be accomplished by requesting content from the most suitable multimedia source.[1] Since both the content generated by the multimedia sources and the user's characteristics change over time, it is unknown to the CA which multimedia source to match with the user. This problem can be formulated as an online learning problem, where the CA learns the best matching by exploring matchings of users with different content providers. After a particular content matching is made, the user "consumes" the content, and provides feedback/rating, such as *like* or *dislike*.[2] It is this feedback that helps a CA learn the preferences of its users and the characteristics of the content that is provided by the multimedia sources. Since this is a learning problem we equivalently call a CA, a content

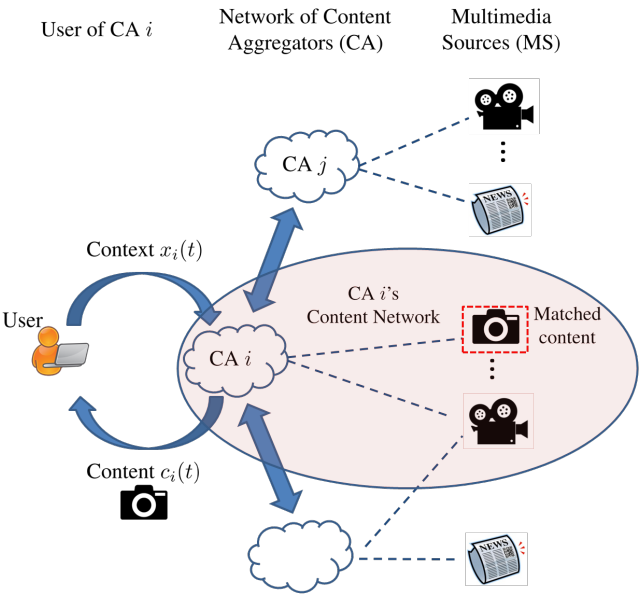

Fig. 1. Operation of the distributed content aggregation system. (i) A user with type/context $x_i(t)$ arrives to Content Aggregator (CA) i, (ii) CA i chooses a *matching action* (either requests content from another CA or requests content from a multimedia source in its own network).

This work is partially supported by the grants NSF CNS 1016081 and AFOSR DDDAS.

C. Tekin and Mihaela van der Schaar are in Department of Electrical Engineering, UCLA, Los Angeles, CA, 90095. Email: cmtkn@ucla.edu, mihaela@ee.ucla.edu.

[1] Although we use the term *request* to explain how content from a multimedia source is mined, our proposed method works also when a CA extracts the content from the multimedia source, without any decision making performed by the multimedia source.

[2] Our framework also works when the feedback is missing for some users.

learner or simply, a *learner*.

Two possible real-world applications of content aggregation are business news aggregation and music aggregation. Business news aggregators can collect information from a variety of multinational and multilingual sources and make personalized recommendations to specific individuals/companies based on their unique needs (see e.g. [7]). Music aggregators enable matching listeners with music content they enjoy both within the content network of the listeners as well as outside this network. For instance, distributed music aggregators can facilitate the sharing of music collections owned by diverse users without the need for centralized content manager/moderator/providers (see e.g. [8]). A discussion of how these applications can be modeled using our framework is given in Section III. Moreover, our proposed methods are tested on real-world datasets related to news aggregation and music aggregation in Section VII.

For each CA i, there are two types of users: *direct* and *indirect*. Direct users are the users that visit the website of CA i to search for content. *Indirect* users are the users of another CA that requests content from CA i. A CA's goal is to maximize the number of likes received from its users (both direct and indirect). This objective can be achieved by all CAs by the following *distributed* learning method: all CAs learn online which *matching action* to take for its current user, i.e., obtain content from a multimedia source that is directly connected, or request content from another CA. However, it is not trivial how to use the past information collected by the CAs in an efficient way, due to the vast number of contexts (different user types) and dynamically changing user and content characteristics. For instance, a certain type of content may become popular among users at a certain point in time, which will require the CA to obtain content from the multimedia source that generates that type of content.

To jointly optimize the performance of the multimedia content aggregation system, we propose an online learning methodology that builds on contextual bandits [9], [10]. The performance of the proposed methodology is evaluated using the notion of regret: the difference between the expected total reward (number of content likes minus costs of obtaining the content) of the best content matching strategy given complete knowledge about the user preferences and content characteristics and the expected total reward of the algorithm used by the CAs. When the user preferences and content characteristics are static, our proposed algorithms achieve sublinear regret in the number of users that have arrived to the system.[3] When the user preferences and content characteristics are slowly changing over time, our proposed algorithms achieve ϵ time-averaged regret, where $\epsilon > 0$ depends on the rate of change of the user and content characteristics.

The remainder of the paper is organized as follows. In Section II, we describe the related work and highlight the differences from our work. In Section III, we describe the decentralized content aggregation problem, the optimal content matching scheme given the complete system model, and the regret of a learning algorithm with respect to the optimal content matching scheme. Then, we consider the model with unknown, static user preferences and content characteristics and propose a distributed online learning algorithm in Section IV. The analysis of the unknown, dynamic user preferences and content characteristics are given in Section VI. Using real-world datasets, we provide numerical results on the performance of our distributed online learning algorithms in Section VII. Finally, the concluding remarks are given in Section VIII.

[3]We use index t to denote the number of users that have arrived so far. We also call t the time index, and assume that one user arrives at each time step.

II. Related Work

Related work can be categorized into two: related work on recommender systems and related work on online learning methods called *multi-armed bandits*.

A. Related work on recommender systems and content matching

A recommender system recommends items to its users based on the characteristics of the users and the items. The goal of a recommender system is to learn which users like which items, and recommend items such that the number of likes is maximized. For instance, in [4], [11] a recommender system that learns the preferences of its users in an online way based on the ratings submitted by the users is provided. It is assumed that the true relevance score of an item for a user is a linear function of the context of the user and the features of the item. Under this assumption, an online learning algorithm is proposed. In contrast, we consider a different model, where the relevance score need not be linear in the context. Moreover, due to the distributed nature of the problem we consider, our online learning algorithms need an additional phase called the training phase, which accounts for the fact that the CAs are uncertain about the information of the other aggregators that they are linked with. We focus on the long run performance and show that the regret per unit time approaches zero when the user and content characteristics are static. An online learning algorithm for a centralized recommender which updates its recommendations as both the preferences of the users and the characteristics of items change over time is proposed in [12].

The general framework which exploits the similarities between the past users and the current user to recommend content to the current user is called collaborative filtering [13]–[15]. These methods find the similarities between the current user and the past users by examining their search and feedback patterns, and then based on the interactions with the past *similar* users, matches the user with the content that has the highest estimated relevance score. For example, the most relevant content can be the content that is liked the highest number of times by similar users. Groups of similar users can be created by various methods such as clustering [14], and then, the matching will be made based on the content matched with the past users that are in the same group.

The most striking difference between our content matching system and previously proposed is that in prior works, there is a central CA which knows the entire set of different types of content, and all the users arrive to this central CA. In contrast,

	Our work	[4], [11]	[14]	[13]	[18]
Distributed	Yes	No	No	No	No
Reward model	Hölder	Linear	N/A	N/A	N/A
Confidence bounds	Yes	No	No	No	No
Regret bound	Yes	Yes	No	No	No
Dynamic user /content distribution	Yes	No	Yes	Yes	Yes

TABLE I
COMPARISON OF OUR WORK WITH OTHER WORK IN RECOMMENDER SYSTEMS

we consider a decentralized system consisting of many CAs, many multimedia sources that these CAs are connected to, and heterogeneous user arrivals to these CAs. These CAs are cooperating with each other by only knowing the connections with their own neighbors but not the entire network topology. Hence, a CA does not know which multimedia sources another CA is connected to, but it learns over time whether that CA has access to content that the users like or not. Thus, our model can be viewed as a giant collection of individual CAs that are running in parallel.

Another line of work [16], [17] uses social streams mined in one domain, e.g., Twitter, to build a topic space that relates these streams to content in the multimedia domain. For example, in [16], Tweet streams are used to provide video recommendations in a commercial video search engine. A content adaptation method is proposed in [6] which enables the users with different types of contexts and devices to receive content that is in a suitable format to be accessed. Video popularity prediction is studied in [17], where the goal is to predict if a video will become popular in the multimedia domain, by detecting social trends in another social media domain (such as Twitter), and transferring this knowledge to the multimedia domain. Although these methods are very different from our methods, the idea of transferring knowledge from one multimedia domain to another can be carried out by CAs specialized in specific types of cross-domain content matching For instance, one CA may transfer knowledge from tweets to predict the content which will have a high relevance/popularity for a user with a particular context, while another CA may scan through the Facebook posts of the user's friends to calculate the context of the domain in addition to the context of the user, and provide a matching according to this.

The advantages of our proposed approach over prior work in recommender systems are: (i) systematic analysis of recommendations' performance, including confidence bounds on the accuracy of the recommendations; (ii) no need for a priori knowledge of the users' preferences (i.e., system learns on-the-fly); (iii) achieve high accuracy even when the users' characteristics and content characteristics are changing over time; (iv) all these features are enabled in a network of distributed CAs.

The differences of our work from the prior work in recommender systems is summarized in Table I.

B. Related work on multi-armed bandits

Other than distributed content recommendation, our learning framework can be applied to any problem that can be formulated as a decentralized contextual bandit problem. Contextual bandits have been studied before in [9], [10], [19]–[21] in a single agent setting, where the agent sequentially chooses from a set of alternatives with unknown rewards, and the rewards depend on the context information provided to the agent at each time step. In [4], a contextual bandit algorithm named LinUCB is proposed for recommending personalized news articles, which is variant of the UCB algorithm [22] designed for linear payoffs. Numerical results on real-world Internet data are provided, but no theoretical results on the resulting regret are derived. The main difference of our work from single agent contextual bandits is that: (i) a three phase learning algorithm with *training*, *exploration* and *exploitation* phases is needed instead of the standard two phase, i.e., *exploration* and *exploitation* phases, algorithms used in centralized contextual bandit problems; (ii) the adaptive partitions of the context space should be formed in a way that each learner/aggregator can efficiently utilize what is learned by other learners about the same context; (iii) the algorithm is robust to missing feedback (some users do not rate the content).

III. PROBLEM FORMULATION

The system model is shown in Fig. 1. There are M content aggregators (CAs) which are indexed by the set $\mathcal{M} := \{1, 2, \ldots, M\}$. We also call each CA a *learner* since it needs to learn which type of content to provide to its users. Let $\mathcal{M}_{-i} := \mathcal{M} - \{i\}$ be the set of CAs that CA i can choose from to request content. Each CA has access to the contents over its content network as shown in Fig. 1. The set of contents in CA i's content network is denoted by $\mathcal{C}_i$. The set of all contents is denoted by $\mathcal{C} := \cup_{i \in \mathcal{M}} \mathcal{C}_i$. The system works in a discrete time setting $t = 1, 2, \ldots, T$, where the following events happen sequentially, in each time slot: (i) a user with context $x_i(t)$ arrives to each CA $i \in \mathcal{M}$,[4] (ii) based on the context of its user each CA matches its user with a content (either from its own content network or by requesting content from another CA), (iii) the user provides a feedback, denoted by $y_i(t)$, which is either *like* ($y_i(t) = 1$) or *dislike* ($y_i(t) = 0$).

The set of *content matching actions* of CA i is denoted by $\mathcal{K}_i := \mathcal{C}_i \cup \mathcal{M}_{-i}$. Let $\mathcal{X} = [0, 1]^d$ be the context space,[5] where d is the dimension of the context space. The context can include many properties of the user such as age, gender, income, previously liked content, etc. We assume that all these quantities are mapped into $[0, 1]^d$. For instance, this mapping can be established by feature extraction methods such as the one given in [4]. Another method is to represent each property of a user by a real number between $[0, 1]$ (e.g., normalize the age by a maximum possible age, represent gender by set $\{0, 1\}$, etc.), without feature extraction. The feedback set of a user is denoted by $\mathcal{Y} := \{0, 1\}$. Let $C_{\max} := \max_{i \in \mathcal{M}} |\mathcal{C}_i|$. We assume that all CAs know $C_{\max}$ but they do not need to know the content networks of other CAs.

The following two examples demonstrate how business news aggregation and music aggregation fits our problem formulation.

Example 1: **Business news aggregation**. Consider a distributed set of news aggregators that operate in different

[4]Although in this model user arrivals are synchronous, our framework will work for asynchronous user arrivals as well.

[5]In general, our results will hold for any bounded subspace of $\mathbb{R}^n$.

countries (for instance a European news aggregator network as in [7]). Each news aggregator's content network (as portrayed in Fig. 1 of the manuscript) consists of content producers (multimedia sources) that are located in specific regions/countries. Consider a user with context x (e.g. age, gender, nationality, profession) who subscribes to the CA A, which is located in the country where the user lives. This CA has access to content from local producers in that country but it can also request content from other CAs, located in different countries. Hence, a CA has access to (local) content generated in other countries. In such scenarios, our proposed system is able to recommend to the user subscribing to CA A also content from other CAs, by discovering the content that is most relevant to that user (based on its context x) across the entire network of CAs. For instance, for a user doing business in the transportation industry, our content aggregator system may learn to recommend road construction news, accidents or gas prices from particular regions that are on the route of the transportation network of the user.

Example 2: **Music aggregation**. Consider a distributed set of music aggregators that are specialized in specific genres of music: classical, jazz, rock, rap, etc. Our proposed model allows music aggregators to share content to provide personalized recommendation for a specific user. For instance, a user that subscribes (frequents/listens) to the classical music aggregator may also like specific jazz tracks. Our proposed system is able to discover and recommend to that user also other music that it will enjoy in addition to the music available to/owned by in aggregator to which it subscribes.

A. User and content characteristics

In this paper we consider two types of user and content characteristics. First, we consider the case when the user and content characteristics are *static*, i.e., they do not change over time. For this case, for a user with context x, $\pi_c(x)$ denotes the probability that the user will like content c. We call this the *relevance score* of content c.

The second case we consider corresponds to the scenario when the characteristics of the users and content are *dynamic*. For online multimedia content, especially for social media, it is known that both the user and the content characteristics are dynamic and noisy [23], hence the problem exhibits concept drift [24]. Formally, a concept is the distribution of the problem, i.e., the joint distribution of the user and content characteristics, at a certain point of time [25]. Concept drift is a change in this distribution. For the case with concept drift, we propose a learning algorithm that takes into account the speed of the drift to decide what window of past observations to use in estimating the relevance score. The proposed learning algorithm has theoretical performance guarantees in contrast to prior work on concept drift which mainly deal with the problem in a ad-hoc manner. Indeed, it is customary to assume that online content is highly dynamic. A certain type of content may become popular for a certain period of time, and then, its popularity may decrease over time and a new content may emerge as popular. In addition, although the type of the content remains the same, such as soccer news, its popularity may change over time due to exogenous events such as the World Cup etc. Similarly, a certain type of content may become popular for a certain type of demographics (e.g., users of a particular age, gender, profession, etc.). However, over time the interest of these users may shift to other types of content. In such cases, where the popularity of content changes over time for a user with context x, $\pi_c(x,t)$ denotes the probability that the user at time t will like content c.

As we stated earlier, a CA i can either recommend content from multimedia sources that it is directly connected to or can ask another CA for content. By asking for content c from another CA j, CA i will incur cost $d_j^i \geq 0$. For the purpose of our paper, the cost is a *generic* term. For instance, it can be a payment made to CA j to display it to CA i's user, or it may be associated with the advertising loss CA i incurs by directing its user to CA j's website. When the cost is payment, it can be money, tokens [26] or Bitcoins [27]. Since this cost is bounded, without loss of generality we assume that $d_j^i \in [0,1]$ for all $i,j \in \mathcal{M}$. In order make our model general, we also assume that there is a cost associated with recommending a type of content $c \in \mathcal{C}_i$, which is given by $d_c^i \in [0,1]$, for CA i. For instance, this can be a payment made to the multimedia source that owns content c.

An intrinsic assumption we make is that the CAs are cooperative. That is, CA $j \in \mathcal{M}_{-i}$ will return the content that is mostly to be liked by CA i's user when asked by CA i to recommend a content. This cooperative structure can be justified as follows. Whenever a user likes the content of CA j (either its own user or user of another CA), CA j obtains a benefit. This can be either an additional payment made by CA i when the content recommended by CA j is liked by CA i's user, or it can simply be the case that whenever a content of CA j is liked by someone its popularity increases. However, we assume that the CAs' decisions do not change their pool of users. The future user arrivals to the CAs are independent of their past content matching strategies. For instance, users of a CA may have monthly or yearly subscriptions, so they will not shift from one CA to another CA when they like the content of the other CA.

The goal of CA i is to explore the matching actions in $\mathcal{K}_i$ to learn the best content for each context, while at the same time exploiting the best content for the user with context $x_i(t)$ arriving at each time instance t to maximize its total number of likes minus costs. CA i's problem can be modeled as a contextual bandit problem [9], [20], [21], [28], where likes and costs translate into rewards. In the next subsection, we formally define the benchmark solution which is computed using perfect knowledge about the probability that a content c will be liked by a user with context x (which requires complete knowledge of user and content characteristics). Then, we define the regret which is the performance loss due to uncertainty about the user and content characteristics.

B. Optimal content matching with complete information

Our benchmark when evaluating the performance of the learning algorithms is the optimal solution which always recommends the content with the highest relevance score minus cost for CA i from the set $\mathcal{C}$ given context $x_i(t)$ at time t. This corresponds to selecting the best matching action in $\mathcal{K}_i$ given

$x_i(t)$. Next, we define the expected rewards of the matching actions, and the *action selection* policy of the benchmark. For a matching action $k \in \mathcal{M}_{-i}$, its relevance score is given as $\pi_k(x) := \pi_{c_k^*(x)}(x)$, where $c_k^*(x) := \arg\max_{c \in \mathcal{C}_j} \pi_c(x)$. For a matching action $k \in \mathcal{C}_i$ its relevance score is equal to the relevance score of content k. The expected reward of CA i from choosing action $k \in \mathcal{K}_i$ is given by the quasilinear utility function

$$\mu_k^i(x) := \pi_k(x) - d_k^i, \tag{1}$$

where $d_k^i \in [0,1]$ is the normalized cost of choosing action k for CA i. Our proposed system will also work for more general expected reward functions as long as the expected reward of a learner is a function of the relevance score of the chosen action and the cost (payment, communication cost, etc.) associated with choosing that action.

The *oracle* benchmark is given by

$$k_i^*(x) := \arg\max_{k \in \mathcal{K}_i} \mu_k^i(x), \quad \forall x \in \mathcal{X}. \tag{2}$$

The oracle benchmark depends on relevance scores as well as costs of matching content from its own content network or other CA's content network. The case $d_k^i = 0$ for all $k \in \mathcal{K}_i$ and $i \in \mathcal{M}$, corresponds to the scheme in which content matching has zero cost, hence $k_i^*(x) = \arg\max_{k \in \mathcal{K}_i} \pi_k(x) = \arg\max_{c \in \mathcal{C}} \pi_c(x)$. This corresponds to the best centralized solution, where CAs act as a single entity. On the other hand, when $d_k^i \geq 1$ for all $k \in \mathcal{M}_{-i}$ and $i \in \mathcal{M}$, in the oracle benchmark a CA must not cooperate with any other CA and should only use its own content. Hence $k_i^*(x) = \arg\max_{c \in \mathcal{C}_i}(\pi_c(x) - d_c^i)$. In the following subsections, we will show that independent of the values of relevance scores and costs, our algorithms will achieve sublinear regret (in the number of users or equivalently time) with respect to the oracle benchmark.

C. The regret of learning

In this subsection we define the regret as a performance measure of the learning algorithm used by the CAs. Simply, the regret is the loss incurred due to the unknown system dynamics. Regret of a learning algorithm which selects the matching action/arm $a_i(t)$ at time t for CA i is defined with respect to the best matching action $k_i^*(x)$ given in (2). Then, the regret of CA i at time T is

$$R_i(T) := \sum_{t=1}^{T} \left(\pi_{k_i^*(x_i(t))}(x_i(t)) - d^i_{k_i^*(x_i(t))} \right) - \mathrm{E}\left[\sum_{t=1}^{T} \left(\mathrm{I}(y_i(t) = L) - d^i_{a_i(t)} \right) \right]. \tag{3}$$

Regret gives the convergence rate of the total expected reward of the learning algorithm to the value of the optimal solution given in (2). Any algorithm whose regret is sublinear, i.e., $R_i(T) = O(T^\gamma)$ such that $\gamma < 1$, will converge to the optimal solution in terms of the average reward.

A summary of notations is given in Table II. In the next section, we propose an online learning algorithm which achieves sublinear regret when the user and content characteristics are static.

$\mathcal{M}$: Set of all CAs.
$\mathcal{C}_i$: Contents in the Content Network of CA i.
$C_{\max}$: $\max_{i \in \mathcal{M}} \lvert\mathcal{C}_i\rvert$.
$\mathcal{C}$: Set of all contents.
$\mathcal{X} = [0,1]^d$: Context space.
$\mathcal{Y}$: Set of feedbacks a user can give.
$x_i(t)$: d-dimensional context of tth user of CA i.
$y_i(t)$: Feedback of the tth user of CA i.
$\mathcal{K}_i$: Set of content matching actions of CA i.
$\pi_c(x)$: Relevance score of content c for context x.
d_k^i: Cost of choosing matching action k for CA i.
$\mu_k^i(x)$: Expected reward (static) of CA i from matching action k for context x.
$k_i^*(x)$: Optimal matching action of CA i given context x (oracle benchmark).
$R_i(T)$: Regret of CA i at time T.
$\beta_a := \sum_{t=1}^{\infty} 1/t^a$

TABLE II
NOTATIONS USED IN PROBLEM FORMULATION.

IV. A Distributed Online Content Matching Algorithm

In this section we propose an online learning algorithm for content matching when the user and content characteristics are static. In contrast to prior online learning algorithms that exploit the context information [9], [10], [19]–[21], [28], which consider a single learner setting, the proposed algorithm helps a CA to learn from the *experience* of other CAs. With this mechanism, a CA is able to recommend content from multimedia sources that it has no direct connection, without needing to know the IDs of such multimedia sources and their content. It learns about these multimedia sources only through the other CAs that it is connected to.

In order to bound the regret of this algorithm analytically we use the following assumption. When the content characteristics are static, we assume that each type of content has similar relevance scores for similar contexts; we formalize this in terms of a Lipschitz condition.

Assumption 1: For each $c \in \mathcal{C}$, there exists $L > 0$, $\gamma > 0$ such that for all $x, x' \in \mathcal{X}$, we have $|\pi_c(x) - \pi_c(x')| \leq L||x - x'||^\gamma$.

Assumption 1 indicates that the probability that a type c content is liked by users with similar contexts will be similar to each other. For instance, if two users have similar age, gender, etc., then it is more likely that they like the same content. We call L the *similarity constant* and γ the *similarity exponent*. These parameters will depend on the characteristics of the users and the content. We assume that γ is known by the CAs. However, an unknown γ can be estimated online using the history of likes and dislikes by users with different contexts, and our proposed algorithms can be modified to include the estimation of γ.

In view of this assumption, the important question becomes how to learn from the past experience which content to match with the current user. We answer this question by proposing an algorithm which partitions the context space of a CA, and learns the relevance scores of different types of content for each set in the partition, based only on the past experience in that set. The algorithm is designed in a way to achieve optimal tradeoff between the size of the partition and the past observations that can be used together to learn the

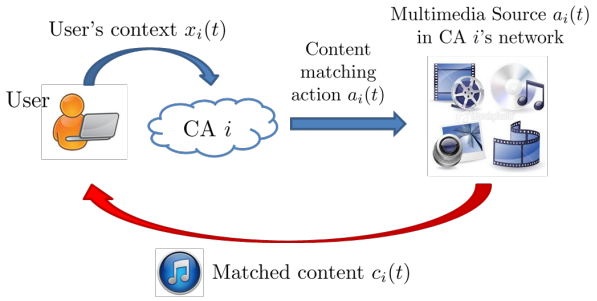

Fig. 2. Content matching within own content network.

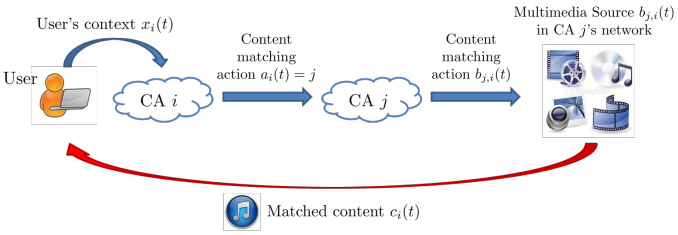

Fig. 3. Content matching from the network of another CA.

relevance scores. It also includes a mechanism to help CAs learn from each other's users. We call our proposed algorithm the *DIStributed COntent Matching algorithm* (DISCOM), and its pseudocode is given in Fig. 4, Fig. 5 and Fig. 6.

Each CA i has two tasks: matching content with its own users and matching content with the users of other CAs when requested by those CAs. We call the first task the *maximization task* (implemented by DISCOM$_{\text{max}}$ given in Fig. 5), since the goal of CA i is to maximize the number of likes from its own users. The second task is called the *cooperation task* (implemented by DISCOM$_{\text{coop}}$ given in Fig. 6), since the goal of CA i is to help other CAs obtain content from its own content network in order to maximize the likes they receive from their users. This cooperation is beneficial to CA i because of numerous reasons. Firstly, since every CA cooperates, CA i can reach a much larger set of content including the content from other CA's content networks, hence will be able to provide content with higher relevance score to its users. Secondly, when CA i helps CA j, it will observe the feedback of CA j's user for the matched content, hence will be able to update the estimated relevance score of its content, which is beneficial if a user similar to CA j's user arrives to CA i in the future. Thirdly, payment mechanisms can be incorporated to the system such that CA i gets a payment from CA j when its content is liked by CA j's user.

In summary, there are two types of content matching actions for a user of CA i. In the first type, the content is recommended from a source that is directly connected to CA i, while in the second type, the content is recommended from a source that CA i is connected through another CA. The information exchange between multimedia sources and CAs for these two types of actions is shown in Fig. 2 and Fig. 3.

Let T be the time horizon of interest (equivalent to the number of users that arrive to each CA). DISCOM creates a partition of $\mathcal{X} = [0,1]^d$ based on T. For instance T can be the average number of visits to the CA's website in one day. Although in reality the average number of visits to different CAs can be different, our analysis of the regret in this section will hold since it is the worst-case analysis (assuming that users arrive only to CA i, while the other CAs only learn through CA i's users). Moreover, the case of heterogeneous number of visits can be easily addressed if each CA informs other CAs about its average number of visits. Then, CA i can keep M different partitions of the context space; one for itself and $M-1$ for the other CAs. If called by CA j, it will match a content to CA j's user based on the partition it keeps for CA j. Hence, we focus on the case when T is common to all CAs.

We first define m_T as the *slicing level* used by DISCOM, which is an integer that is used to partition $\mathcal{X}$. DISCOM forms a partition of $\mathcal{X}$ consisting of $(m_T)^d$ sets (hypercubes) where each set is a d-dimensional hypercube with edge length $1/m_T$. This partition is denoted by $\mathcal{P}_T$. The hypercubes in $\mathcal{P}_T$ are oriented such that one of them has a corner located at the origin of the d-dimensional Euclidian space. It is clear that the number of hypercubes is increasing in m_T, while their size is decreasing in m_T. When m_T is small each hypercube covers a large set of contexts, hence the number of past observations that can be used to estimate relevance scores of matching actions in each set is large. However, the variation of the true relevance scores of the contexts within a hypercube increases with the size of the hypercube. DISCOM should set m_T to a value that balances this tradeoff.

A hypercube in $\mathcal{P}_T$ is denoted by p. The hypercube in $\mathcal{P}_T$ that contains context $x_i(t)$ is denoted by $p_i(t)$. When $x_i(t)$ is located at a boundary of multiple hypercubes in $\mathcal{P}_T$, it is randomly assigned to one of these hypercubes.

DISCOM$_{\text{max}}$ operates as follows. CA i matches its user at time t with a content by taking a matching action based on one of the three phases: *training* phase in which CA i requests content from another CA j for the purpose of helping CA j to learn the relevance score of content in its content network for users with context $x_i(t)$ (but CA i does not update the relevance score for CA j because it thinks that CA j may not know much about its own content), *exploration* phase in which CA i selects a matching action in $\mathcal{K}_i$ and updates its relevance score based on the feedback of its user, and *exploitation* phase in which CA i chooses the matching action with the highest relevance score minus cost.

Since the CAs are cooperative, when another CA requests content from CA i, CA i will choose content from its content network with the highest estimated relevance score for the user of the requesting CA. To maximize the number of likes minus costs in exploitations, CA i must have an accurate estimate of the relevance scores of other CAs. This task is not trivial since CA i does not know the content network of other CAs. In order to do this, CA i should smartly select which of its users' feedbacks to use when estimating the relevance score of CA j. The feedbacks should come from previous times at which CA i has a very high confidence that the content of CA j matched with its user is the one with the highest relevance score for the context of CA i's user. Thus, the training phase of CA i helps other CAs build accurate estimates about the relevance scores of their content, before CA i uses any feedback for content coming from these CAs to form relevance score estimates

DISCOM for CA i:

Input: $H_1(t)$, $H_2(t)$, $H_3(t)$, T, m_T
Initialize: Partition $\mathcal{X}$ into hypercubes denoted by $\mathcal{P}_T$.
Initialize: Set counters $N_p^i = 0$, $\forall p \in \mathcal{P}_T$, $N_{k,p}^i = 0, \forall k \in \mathcal{K}_i, p \in \mathcal{P}_T$, $N_{j,p}^{\text{tr},i} = 0, \forall j \in \mathcal{M}_{-i}, p \in \mathcal{P}_T$.
Initialize: Set relevance score estimates $\bar{r}_{k,p}^i = 0$, $\forall k \in \mathcal{K}_i$, $p \in \mathcal{P}_T$.
while $t \geq 1$ **do**
  Run DISCOM$_{\text{max}}$ to find $p = p_i(t)$, to obtain a matching action a_i, and value of $train$ flag.
  If $a_i \in \mathcal{M}_{-i}$ ask CA a_i for content and pass $x_i(t)$.
  Receive $\mathcal{CA}_i(t)$, the set of CAs who requested content from CA i, and their contexts.
  if $\mathcal{CA}_i(t) \neq \emptyset$ **then**
    Run DISCOM$_{\text{coop}}$ to obtain the content to be selected $\boldsymbol{b}_i := \{b_{i,j}\}_{j \in \mathcal{CA}_i(t)}$ and hypercubes that the contexts of the users in $\mathcal{CA}_i(t)$ lie in $\boldsymbol{p}_i := \{p_{i,j}\}_{j \in \mathcal{CA}_i(t)}$.
  end if
  if $a_i \in \mathcal{C}_i$ **then**
    Pay cost $d_{a_i}^i$, obtain content a_i.
    Show a_i to the user, receive feedback $r \in \{0,1\}$ drawn from $\text{Ber}_{a_i}(x_i(t))$.[a]
  else
    Pay cost $d_{a_i}^i$, obtain content $b_{a_i,i}$ from CA a_i.
    Show $b_{a_i,i}$ to the user, receive feedback $r \in \{0,1\}$ drawn from $\text{Ber}_{b_{a_i,i}}(x_i(t))$.
  end if
  if $train = 1$ **then**
    $N_{a_i,p}^{\text{tr},i}++$.
  else
    $\bar{r}_{a_i,p}^i = (\bar{r}_{a_i,p}^i N_{a_i,p}^i + r)/(N_{a_i,p}^i + 1)$.
    N_p^i++, $N_{a_i,p}^i++$.
  end if
  if $\mathcal{CA}_i(t) \neq \emptyset$ **then**
    for $j \in \mathcal{CA}_i(t)$ **do**
      Send content $b_{i,j}$ to CA j's user.
      Observe feedback r drawn from $\text{Ber}_{b_{i,j}}(x_j(t))$.
      $\bar{r}_{b_{i,j},p_{i,j}}^i = \frac{\bar{r}_{b_{i,j},p_{i,j}}^i N_{b_{i,j},p_{i,j}}^i + r}{N_{b_{i,j},p_{i,j}}^i + 1}$.
      $N_{p_{i,j}}^i++$, $N_{b_{i,j},p_{i,j}}^i++$.
    end for
  end if
  $t = t + 1$
end while

[a] $\text{Ber}_{a_i}(x_i(t))$ is the Bernoulli distribution with expected value $\pi_{a_i}(x_i(t))$.

Fig. 4. Pseudocode for DISCOM algorithm.

about them. In contrast, the exploration phase of CA i helps it to build accurate estimates about the relevance score of its matching actions.

At time t, the phase that CA i will be in is determined by the amount of time it had explored, exploited or trained for past users with contexts similar to the context of the current user. For this CA i keeps counters and control functions which are described below. Let $N_p^i(t)$ be the number of user arrivals to CA i with contexts in $p \in \mathcal{P}_T$ by time t (its own arrivals and arrivals to other CAs who requested content from CA i) except the training phases of CA i. For $c \in \mathcal{C}_i$, let $N_{c,p}^i(t)$ be the number of times content c is selected in response to a user arriving to CA i with context in hypercube p by time t (including times other CAs request content from CA i for their users with contexts in set p). Other than these, CA i keeps two counters for each other CA in each set in the partition, which it uses to decide the phase it should be in. The first one, i.e., $N_{j,p}^{\text{tr},i}(t)$, is an estimate on the number of user arrivals with contexts in p to CA j from all CAs except the training phases

DISCOM$_{\text{max}}$ (maximization part of DISCOM) for CA i:

$train = 0$.
Find the hypercube in $\mathcal{P}_T$ that $x_i(t)$ belongs to, i.e., $p_i(t)$.
Let $p = p_i(t)$.
Compute the set of under-explored matching actions $\mathcal{C}_{i,p}^{\text{ue}}(t)$ given in (4).
if $\mathcal{C}_{i,p}^{\text{ue}}(t) \neq \emptyset$ **then**
  Select a_i randomly from $\mathcal{C}_{i,p}^{\text{ue}}(t)$.
else
  Compute the set of training candidates $\mathcal{M}_{i,p}^{\text{ct}}(t)$ given in (5).
  //Update the counters of training candidates.
  for $j \in \mathcal{M}_{i,p}^{\text{ut}}(t)$ **do**
    Obtain N_p^j from CA j, set $N_{j,p}^{\text{tr},i} = N_p^j - N_{j,p}^i$.
  end for
  Compute the set of under-trained CAs $\mathcal{M}_{i,p}^{\text{ut}}(t)$ given in (6).
  Compute the set of under-explored CAs $\mathcal{M}_{i,p}^{\text{ue}}(t)$ given in (7).
  if $\mathcal{M}_{i,p}^{\text{ut}}(t) \neq \emptyset$ **then**
    Select a_i randomly from $\mathcal{M}_{i,p}^{\text{ut}}(t)$, $train = 1$.
  else if $\mathcal{M}_{i,p}^{\text{ue}}(t) \neq \emptyset$ **then**
    Select a_i randomly from $\mathcal{M}_{i,p}^{\text{ue}}(t)$.
  else
    Select a_i randomly from $\arg\max_{k \in \mathcal{K}_i} \bar{r}_{k,p}^i - d_k^i$.
  end if
end if

Fig. 5. Pseudocode for the maximization part of DISCOM algorithm.

DISCOM$_{\text{coop}}$ (cooperation part of DISCOM) for CA i.

for $j \in \mathcal{CA}_i(t)$ **do**
  Find the set in $\mathcal{P}_T$ that $x_j(t)$ belongs to, i.e., $p_{i,j}$.
  Compute the set of under-explored matching actions $\mathcal{C}_{i,p_{i,j}}^{\text{ue}}(t)$ given in (4).
  if $\mathcal{C}_{i,p_{i,j}}^{\text{ue}}(t) \neq \emptyset$ **then**
    Select $b_{i,j}$ randomly from $\mathcal{C}_{i,p_{i,j}}^{\text{ue}}(t)$.
  else
    $b_{i,j} = \arg\max_{c \in \mathcal{C}_i} \bar{r}_{c,p_{i,j}}^i$.
  end if
end for

Fig. 6. Pseudocode for the cooperation part of DISCOM algorithm.

of CA j and exploration, exploitation phases of CA i. This counter is only updated when CA i thinks that CA j should be trained. The second one, i.e., $N_{j,p}^i(t)$, counts the number of users of CA i with contexts in p for which content is requested from CA j at exploration and exploitation phases of CA i by time t.

At each time slot t, CA i first identifies $p_i(t)$. Then, it chooses its phase at time t by giving highest priority to exploration of content in its own content network, second highest priority to training of the other CAs, third highest priority to exploration of the other CAs, and lowest priority to exploitation. The reason that exploration of own content has a higher priority than training of other CAs is that it will minimize the number of times CA i will be trained by other CAs, which we describe below.

First, CA i identifies the set of under-explored content in its content network:

$$\mathcal{C}_{i,p}^{\text{ue}}(t) := \{c \in \mathcal{C}_i : N_{c,p}^i(t) \leq H_1(t)\}. \tag{4}$$

where $H_1(t)$ is a deterministic, increasing function of t which is called *the control function*. The value of this function will affect the regret of DISCOM. For $c \in \mathcal{C}_i$, the accuracy of relevance score estimates increase with $H_1(t)$, hence it should be selected to balance the tradeoff between accuracy and the number of explorations. If $\mathcal{C}_{i,p}^{\text{ue}}(t)$ is non-empty, CA i

enters the exploration phase and randomly selects a content in this set to explore. Otherwise, it identifies the set of training candidates:

$$\mathcal{M}^{\text{ct}}_{i,p}(t) := \{j \in \mathcal{M}_{-i} : N^{\text{tr},i}_{j,p}(t) \leq H_2(t)\}, \tag{5}$$

where $H_2(t)$ is a control function similar to $H_1(t)$. Accuracy of other CA's relevance score estimates of content in their own networks increases with $H_2(t)$, hence it should be selected to balance the possible reward gain of CA i due to this increase with the reward loss of CA i due to the number of trainings. If this set is non-empty, CA i asks the CAs $j \in \mathcal{M}^{\text{ct}}_{i,p}(t)$ to report $N^j_p(t)$. Based in the reported values it recomputes $N^{\text{tr},i}_{j,p}(t)$ as $N^{\text{tr},i}_{j,p}(t) = N^j_p(t) - N^i_{j,p}(t)$. Using the updated values, CA i identifies the set of under-trained CAs:

$$\mathcal{M}^{\text{ut}}_{i,p}(t) := \{j \in \mathcal{M}_{-i} : N^{\text{tr},i}_{j,p}(t) \leq H_2(t)\}. \tag{6}$$

If this set is non-empty, CA i enters the training phase and randomly selects a CA in this set to train it. When $\mathcal{M}^{\text{ct}}_{i,p}(t)$ or $\mathcal{M}^{\text{ut}}_{i,p}(t)$ is empty, this implies that there is no under-trained CA, hence CA i checks if there is an under-explored matching action. The set of CAs for which CA i does not have accurate relevance scores is given by

$$\mathcal{M}^{\text{ue}}_{i,p}(t) := \{j \in \mathcal{M}_{-i} : N^i_{j,p}(t) \leq H_3(t)\}, \tag{7}$$

where $H_3(t)$ is also a control function similar to $H_1(t)$. If this set is non-empty, CA i enters the exploration phase and randomly selects a CA in this set to request content from to explore it. Otherwise, CA i enters the exploitation phase in which it selects the matching action with the highest estimated relevance score minus cost for its user with context $x_i(t) \in p = p_i(t)$, i.e.,

$$a_i(t) \in \arg\max_{k \in \mathcal{K}_i} \bar{r}^i_{k,p}(t) - d^i_k, \tag{8}$$

where $\bar{r}^i_{k,p}(t)$ is the sample mean estimate of the relevance score of CA i for matching action k at time t, which is computed as follows. For $j \in \mathcal{M}_{-i}$, let $\mathcal{E}^i_{j,p}(t)$ be the set of feedbacks collected by CA i at times it selected CA j while CA i's users' contexts are in set p in its exploration and exploitation phases by time t. For estimating the relevance score of contents in its own content network, CA i can also use the feedback obtained from other CAs' users at times they requested content from CA i. In order to take this into account, for $c \in \mathcal{C}_i$, let $\mathcal{E}^i_{c,p}(t)$ be the set of feedbacks observed by CA i at times it selected its content c for its own users with contexts in set p union the set of feedbacks observed by CA i when it selected its content c for the users of other CAs with contexts in set p who requests content from CA i by time t.

Therefore, sample mean relevance score of matching action $k \in \mathcal{K}_i$ for users with contexts in set p for CA i is defined as $\bar{r}^i_{k,p}(t) = \left(\sum_{r \in \mathcal{E}^i_{k,p}(t)} r\right) / |\mathcal{E}^i_{k,p}(t)|$, An important observation is that computation of $\bar{r}^i_{k,p}(t)$ does not take into account the matching costs. Let $\hat{\mu}^i_{k,p}(t) := \bar{r}^i_{k,p}(t) - d^i_k$ be the estimated *net reward* (relevance score minus cost) of matching action k for set p. Of note, when there is more than one maximizer of (8), one of them is randomly selected. In order to run DISCOM, CA i does not need to keep the sets $\mathcal{E}^i_{k,p}(t)$ in its memory. $\bar{r}^i_{k,p}(t)$ can be computed by using only $\bar{r}^i_{k,p}(t-1)$ and the feedback at time t.

L: Similarity constant. γ: Similarity exponent
T: Time horizon.
m_T: Slicing level of DISCOM.
$\mathcal{P}_T$: DISCOM's partition of $\mathcal{X}$ into $(m_T)^d$ hypercubes.
$p_i(t)$: Hypercube in $\mathcal{P}_T$ that contains $x_i(t)$.
$N^i_p(t)$: Number of *all* user arrivals to CA i with contexts in $p \in \mathcal{P}_T$ by time t except the training phases of CA i.
$N^i_{c,p}(t)$: Number of times content c is selected in response to a user arriving to CA i with context in hypercube p by time t.
$N^{\text{tr},i}_{j,p}(t)$: Estimate of CA i on the number of user arrivals with contexts in p to CA j from all CAs except the training phases of CA j and exploration, exploitation phases of CA i.
$N^i_{j,p}(t)$: Number of users of CA i with contexts in p for which content is requested from CA j at exploration and exploitation phases of CA i by time t.
$H_1(t), H_2(t), H_3(t)$: Control functions of DISCOM.
$\mathcal{C}^{\text{ue}}_{i,p}(t)$: Set of under-explored content in $\mathcal{C}_i$.
$\mathcal{M}^{\text{ct}}_{i,p}(t)$: Set of training candidates of CA i.
$\mathcal{M}^{\text{ut}}_{i,p}(t)$: Set of CAs under-trained by CA i.
$\mathcal{M}^{\text{ue}}_{i,p}(t)$: Set of CAs under-explored by CA i.
$\bar{r}^i_{k,p}(t)$: Sample man relevance score of action k of CA i at time t.
$\hat{\mu}^i_{k,p}(t)$ Estimated net reward of action k of CA i at time t.

TABLE III
NOTATIONS USED IN DEFINITION OF DISCOM.

The cooperation part of DISCOM, i.e., DISCOM$_{\text{coop}}$ operates as follows. Let $\mathcal{CA}_i(t)$ be the set CAs who request content from CA i at time t. For each $j \in \mathcal{CA}_i(t)$, CA i first checks if it has any under-explored content c for $p_j(t)$, i.e., c such that $N^i_{c,p_j(t)}(t) \leq H_1(t)$. If so, it randomly selects one of its under-explored content to match it with the user of CA j. Otherwise, it exploits its content in $\mathcal{C}_i$ with the highest estimated relevance score for CA j's current user's context, i.e.,

$$b_{i,j}(t) \in \arg\max_{c \in \mathcal{C}_i} \bar{r}^i_{c,p_j(t)}(t). \tag{9}$$

A summary of notations used in the description of DISCOM is given in Table III.

The following theorem provides a bound on the regret of DISCOM.

Theorem 1: When DISCOM is run by all CAs with parameters $H_1(t) = t^{2\gamma/(3\gamma+d)} \log t$, $H_2(t) = C_{\max} t^{2\gamma/(3\gamma+d)} \log t$, $H_3(t) = t^{2\gamma/(3\gamma+d)} \log t$ and $m_T = \lceil T^{1/(3\gamma+d)} \rceil$,[6] we have

$$\begin{aligned} R_i(T) &\leq 4(M-1)C_{\max}\beta_2 \\ &+ T^{\frac{2\gamma+d}{3\gamma+d}} \left(\frac{2(2Ld^{\gamma/2}+6)}{(2\gamma+d)/(3\gamma+d)} + 6Ld^{\gamma/2} + 2^{d+1} Z_i \log T \right) \\ &+ T^{\frac{\gamma+d}{3\gamma+d}} \frac{2^{d+3}(M-1)C_{\max}\beta_2}{2\gamma/(3\gamma+d)} \\ &+ T^{\frac{d}{3\gamma+d}} 2^{d+1}(|\mathcal{C}_i|(1+2\beta_2) + 2(M-1)(1+\beta_2)), \end{aligned}$$

i.e., $R_i(T) = \tilde{O}\left(MC_{\max} T^{\frac{2\gamma+d}{3\gamma+d}}\right)$,[7] where $Z_i = |\mathcal{C}_i| + (M-1)(C_{\max}+1)$.

Proof: The proof is given in our online technical report [29]. ∎

[6]For a number $r \in \mathbb{R}$, let $\lceil r \rceil$ be the smallest integer that is greater than or equal to r.

[7]$\tilde{O}(\cdot)$ is the Big-O notation in which the terms with logarithmic growth rates are hidden.

For any $d > 0$ and $\gamma > 0$, the regret given in Theorem 1 is sublinear in time (or number of user arrivals). This guarantees that the regret per-user, i.e., the time averaged regret, converges to 0 ($\lim_{T\to\infty} \mathrm{E}[R_i(T)]/T = 0$). It is also observed that the regret increases in the dimension d of the context. By Assumption 1, a context is similar to another context if they are similar in each dimension, hence number of hypercubes in the partition $\mathcal{P}_T$ increases with d.

In our analysis of the regret of DISCOM we assumed that T is fixed and given as an input to the algorithm. DISCOM can be made to run independently of the final time T by using a standard method called *the doubling trick* (see, e.g., [9]). The idea is to divide time into rounds with geometrically increasing lengths and run a new instance of DISCOM at each round. For instance, consider rounds $\tau \in \{1, 2, \ldots\}$, where each round has length 2^τ. Run a new instance of DISCOM at the beginning of each round with time parameter 2^τ. This modified version will also have $\tilde{O}\left(T^{(2\gamma+d)/(3\gamma+d)}\right)$ regret.

Maximizing the satisfaction of an individual user is as important as maximizing the overall satisfaction of all users. The next corollary shows that by using DISCOM, CAs guarantee that their users will almost always be provided with the best content available within the entire content network.

Corollary 1: Assume that DISCOM is run with the set of parameters given in Theorem 1. When DISCOM is in exploitation phase for CA i, we have

$$P(\mu^i_{a_i(t)}(x_i(t)) \leq \mu^i_{k^*_i(x_i(t))}(x_i(t)) - \delta_T) \leq \frac{2|\mathcal{K}_i|}{t^2} + \frac{2(M-1)C_{\max}\beta_2}{t^{(2\gamma+d)/(3\gamma+d)}},$$

where $\delta_T = (4Ld^{\gamma/2} + 6)T^{-\gamma/(3\gamma+d)}$.

Proof: The proof is given in our online technical report [29]. ∎

Remark 1: **(Differential Services)** Maximizing the performance for an individual user is particularly important for providing *differential services* based on the types of the users. For instance, a CA may want to provide higher quality recommendations to a subscriber (high type user) who has paid for the subscription compared to a non-subscriber (low type user). To do this, the CA can exploit the best content for the subscribed user, while perform exploration on a different user that is not subscribed.

V. Regret When Feedback is Missing

When analyzing the performance of DISCOM, we assumed that the users always provide a feedback: like or dislike. However, in most of the online content aggregation platforms user feedback is not always available. In this section we consider the effect of missing feedback on the performance of the proposed algorithm. We assume that each user gives a feedback with probability p_r (which is unknown to the CAs). If the user at time t does not give feedback, we assume that DISCOM does not update its counters. This will result in a larger number of trainings and explorations compared to the case when feedback is always available. The following theorem gives an upper bound on the regret of DISCOM for this case.

Theorem 2: Let the DISCOM algorithm run with parameters $H_1(t) = t^{2\gamma/(3\gamma+d)} \log t$, $H_2(t) = C_{\max} t^{2\gamma/(3\gamma+d)} \log t$, $H_3(t) = t^{2\gamma/(3\gamma+d)} \log t$, and $m_T = \lceil T^{1/(3\gamma+d)} \rceil$. Then, if a user reveals its feedback with probability p_r, we have for CA i,

$$\begin{aligned} R_i(T) &\leq 4(M-1)C_{\max}\beta_2 \\ &+ T^{\frac{2\gamma+d}{3\gamma+d}} \left(\frac{2(2Ld^{\gamma/2}+6)}{(2\gamma+d)/(3\gamma+d)} + 6Ld^{\gamma/2} + \frac{2^{d+1}Z_i}{p_r} \log T \right) \\ &+ T^{\frac{\gamma+d}{3\gamma+d}} \frac{2^{d+3}(M-1)C_{\max}\beta_2}{2\gamma/(3\gamma+d)} \\ &+ T^{\frac{d}{3\gamma+d}} 2^{d+1}(2Z_i\beta_2 + (|\mathcal{K}_i| + (M-1))/p_r), \end{aligned}$$

i.e., $R_i(T) = \tilde{O}\left(MC_{\max}T^{\frac{2\gamma+d}{3\gamma+d}}/p_r\right)$, where $Z_i = |\mathcal{C}_i| + (M-1)(C_{\max}+1)$, $\beta_a := \sum_{t=1}^{\infty} 1/t^a$.

Proof: The proof is given in our online technical report [29]. ∎

From Theorem 2, we see that missing feedback does not change the time order of the regret. However, the regret is scaled with $1/p_r$, which is the expected number of users required for a single feedback.

VI. Learning Under Dynamic User and Content Characteristics

When the user and content characteristics change over time, the relevance score of content c for a user with context x changes over time. In this section, we assume that the following relation holds between the probabilities that a content will be liked with users with similar contexts at two different times t and t'.

Assumption 2: For each $c \in \mathcal{C}$, there exists $L > 0$, $\gamma > 0$ such that for all $x, x' \in \mathcal{X}$, we have

$$|\pi_{c,t}(x) - \pi_{c,t'}(x')| \leq L\left(||x - x'||\right)^\gamma + |t - t'|/T_s,$$

where $1/T_s > 0$ is the speed of the change in user and content characteristics. We call T_s the *stability parameter*.

Assumption 2 captures the temporal dynamics of content matching which is absent in Assumption 1. Such temporal variations are often referred to as *concept drift* [30], [31]. When there is concept drift, a learner should also consider which past information to take into account when learning, in addition to how to combine the past information to learn the best matching strategy.

The following modification of DISCOM will deal with dynamically changing user and content characteristics by using a time window of past observations in estimating the relevance scores. The modified algorithm is called DISCOM with time window (DISCOM-W). This algorithm groups the time slots into rounds $\zeta = 1, 2, \ldots$ each having a fixed length of $2\tau_h$ time slots, where τ_h is an integer called *the half window length*. Some of the time slots in these rounds overlap with each other as given in Fig. 7. The idea is to keep separate control functions and counters for each round, and calculate the sample mean relevance scores for groups of similar contexts based only on the observations that are made during the time window of that round. We call $\eta = 1$ the *initialization round*. The control functions for the initialization round of DISCOM-W

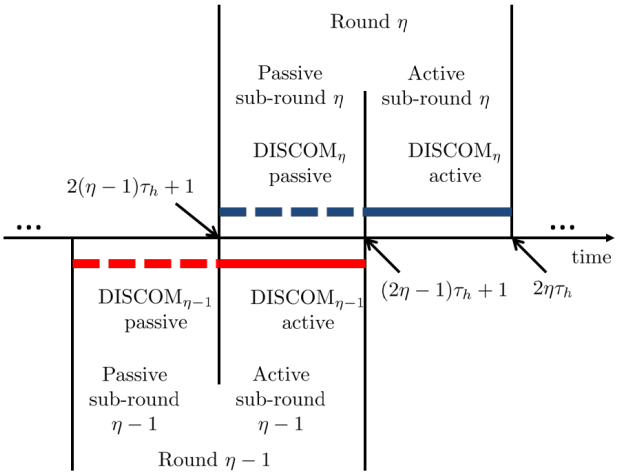

Fig. 7. Operation of DISCOM-W showing the round structure and the different instances of DISCOM running for each round.

is the same as the control functions $H_1(t)$, $H_2(t)$ and $H_3(t)$ of DISCOM whose values are given in Theorem 1. For the other rounds $\zeta > 1$, the control functions depend on τ_h and are given as $H_1^{\tau_h}(t) = H_3^{\tau_h}(t) = (t \mod \tau_h + 1)^z \log(t \mod \tau_h + 1)$ and $H_2^{\tau_h}(t) = C_{\max}(t \mod \tau_h + 1)^z \log(t \mod \tau_h + 1)$, for some $0 < z < 1$. Each round η is divided into two sub-rounds. Except the initialization round, i.e., $\eta = 1$, the first sub-round is called the *passive* sub-round, while the second sub-round is called the *active* sub-round. For the initialization round both sub-rounds are active sub-rounds. In order to reduce the number of trainings and explorations, DISCOM-W has an overlapping round structure as shown in Fig. 7. For each round except the initialization round, passive sub-rounds of round η, overlaps with the active sub-round of round $\eta - 1$. DISCOM-W operates in the same way as DISCOM in each round. DISCOM-W can be viewed as an algorithm which generates a new instance of DISCOM at the beginning of each round, with the modified control functions. DISCOM-W runs two different instances of DISCOM at each round. One of these instances is the active instance based on which content matchings are performed, and the other one is the passive instance which learns through the content matchings made by the active instance.

Let the instance of DISCOM that is run by DISCOM-W at round η be DISCOM$_\eta$. The hypercubes of DISCOM$_\eta$ are formed in a way similar to DISCOM's. The input time horizon is taken as T_s which is the stability parameter given in Assumption 2, and the slicing parameter m_{T_s} is set accordingly. DISCOM$_\eta$ uses the partition of $\mathcal{X}$ into $(m_{T_s})^d$ hypercubes denoted by $\mathcal{P}_{T_s}$. When all CAs are using DISCOM-W, the matching action selection of CA i only depends on the history of content matchings and feedback observations at round η. If time t is in the active sub-round of round η, matching action of CA $i \in \mathcal{M}$ is taken according to DISCOM$_\eta$. As a result of the content matching, sample mean relevance scores and counters of both DISCOM$_\eta$ and DISCOM$_{\eta+1}$ are updated. Else if time t is in the passive sub-round of round η, matching action of CA $i \in \mathcal{M}$ is taken according to DISCOM$_{\eta-1}$ (see Fig. 7). As a result of this, sample mean relevance scores and counters of both DISCOM$_{\eta-1}$ and DISCOM$_\eta$ are updated.

At the start of a round η, the relevance score estimates and counters for DISCOM$_\eta$ are equal to zero. However, due to the two sub-round structure, when the active sub-round of round η starts, CA i already has some observations for the context and actions taken in the passive sub-round of that round, hence depending on the arrivals and actions in the passive sub-round, the CA may even start the active sub-round by exploiting, whereas it should have always spent some time in training and exploration if it starts an active sub-round without any past observations (cold start problem).

In this section, due to the concept drift, even though the context of a past user can be similar to the context of the current user, their relevance scores for a content c can be very different. Hence DISCOM-W assumes that a past user is similar to the current user only if it arrived in the current round. Since round length is fixed, it is impossible to have sublinear number of similar context observations for every t. Thus, achieving sublinear regret under concept drift is not possible. Therefore, in this section we focus on the average regret which is given by

$$R_i^{\text{avg}}(T) := \frac{1}{T}\sum_{t=1}^{T}\left(\pi_{k_i^*(x_i(t))}(x_i(t)) - d^i_{k_i^*(\boldsymbol{x}_i(t))}\right) - \frac{1}{T}\mathrm{E}\left[\sum_{t=1}^{T}\left(\mathrm{I}(y_i(t)=L) - d^i_{a_i(t)}\right)\right].$$

The following theorem bounds the average regret of DISCOM-W.

Theorem 3: When DISCOM-W is run with parameters

$$H_1^{\tau_h}(t) = H_3^{\tau_h}(t) = (t \bmod \tau_h + 1)^{\frac{2\gamma}{3\gamma+d}} \log(t \bmod \tau_h + 1),$$
$$H_2^{\tau_h}(t) = C_{\max}(t \bmod \tau_h + 1)^{\frac{2\gamma}{3\gamma+d}} \log(t \bmod \tau_h + 1),$$

$m_{T_s} = \lceil T_s^{\frac{1}{3\gamma+d}} \rceil$ and $\tau_h = \lfloor T_s^{(3\gamma+d)/(4\gamma+d)} \rfloor$,[8] where T_s is the stability parameter which is given in Assumption 2, the time averaged regret of CA i by time T is $R_i^{\text{avg}}(T) = \tilde{O}\left(T_s^{\frac{-\gamma}{4\gamma+d}}\right)$, for any $T > 0$. Hence DISCOM-W is $\epsilon = \tilde{O}\left(T_s^{\frac{-\gamma}{4\gamma+d}}\right)$ approximately optimal in terms of the average reward.

Proof: The proof is given in our online technical report [29]. ∎

From the result of this theorem we see that the average regret decays as the stability parameter T_s increases. This is because, DISCOM-W will use a longer time window (round) when T_s is large, and thus can get more observations to estimate the sample mean relevance scores of the matching actions in that round, which will result in better estimates hence smaller number of suboptimal matching action selections. Moreover, the average number of trainings and explorations required decrease with the round length.

VII. Numerical Results

In this section we provide numerical results for our proposed algorithms DISCOM and DISCOM-W on real-world datasets.

[8]For a number b, $\lfloor b \rfloor$ denotes the largest integer that is smaller than or equal to b.

A. Datasets

For all the datasets below, for a CA the cost of choosing a content within the content network and the cost of choosing another CA is set to 0. Hence, the only factor that affects the total reward is the users' ratings for the contents.

Yahoo! Today Module (YTM) [4]: The dataset contains news article webpage recommendations of Yahoo! Front Page. Each instance is composed of (i) ID of the recommended content, (ii) the user's context (2-dimensional vector), (iii) the user's click information. The user's click information for a webpage/content is associated with the relevance score of that content. It is equal to 1 if the user clicked on the recommended webpage and 0 else. The dataset contains $T = 70000$ instances and 40 different types of content. We generate 4 CAs and assign 10 of the 40 types of content to each CA's content network. Each CA has direct access to content in its own network, while it can also access to the content in other CAs' content network by requesting content from these CAs. Users are divided into four groups according to their contexts and each group is randomly assigned to one of the CAs. Hence, the user arrival processes to different CA's are different. The performance of a CA is evaluated in terms of the average number of clicks, i.e., click through rate (CTR), of the contents that are matched with its users.

Music Dataset (MD): The dataset contains contextual information and ratings (like/dislike) of music genres (classical, rock, pop, rap) collected from 413 students at UCLA. We generate 2 CAs each specialized in two of the four music genres. Users among the 413 users randomly arrive to each CA. A CA either recommends a music content that is in its content network or asks another CA, specialized in another music genre, to provide a music item. As a result, the rating of the user for the genre of the provided music content is revealed to the CA. The performance of a CA is evaluated in terms of the average number of likes it gets for the contents that are matched with its users.

Yahoo! Today Module (YTM) with Drift (YTMD): This dataset is generated from YTM to simulate the scenario where the user ratings for a particular content changes over time. After every 10000 instances, 20 contents are randomly selected and user clicks for these contents are set to 0 (no click) for the next 10000 instances. For instance, this can represent a scenario where some news articles lose their popularity a day after they become available while some other news articles related to ongoing events will stay popular for several days.

B. Learning Algorithms

While DISCOM and DISCOM-W are the first distributed algorithms to perform content aggregation (see Table I), we compare their performance with distributed versions of the centralized algorithms proposed in [4], [9], [18], [21]. In the distributed implementation of these centralized algorithms, we assume that each CA runs an independent instance of these algorithms. For instance, when implementing a centralized algorithm A on the distributed system of CAs, we assume that each CA i runs its own instance of A denoted by A_i. When CA i selects CA j as a matching action in $\mathcal{K}_i$ by using its algorithm A_i, CA j will select the content for CA i using its algorithm A_j with CA i's user's context on the set of contents $\mathcal{C}_j$. In our numerical results, each algorithm is run for different values of its input parameters. The results are shown for the parameter values for which the corresponding algorithm performs the best.

DISCOM: Our algorithm given in Fig. 4 with control functions $H_1(t)$, $H_2(t)$ and $H_3(t)$ divided by 10 for MD, and by 20 for YTM and YTMD to reduce the number of trainings and explorations.[9]

DISCOM-W: Our algorithm given in Fig. 7 which is the time-windowed version of DISCOM with control functions $H_1(t)$, $H_2(t)$ and $H_3(t)$ divided by 20 to reduce the number of trainings and explorations.

As we mentioned in Remark 1, both DISCOM and DISCOM-W can provide differential services to its users. In this case both algorithms always exploit for the users with high type (subscribers) and if necessary can train and explore for the users with low type (non-subscribers). Hence, the performance of DISCOM and DISCOM-W for differential services is equal to their performance for the set of high type users.

LinUCB [4], [21]: This algorithm computes an index for each matching action by assuming that the relevance score of a matching action for a user is a linear combination of the contexts of the user. Then for each user it selects the matching action with the highest index.

Hybrid-ϵ [18]: This algorithm forms context-dependent sample mean rewards for the matching actions by considering the history of observations and decisions for groups of contexts that are similar to each other. For user t it either explores a random matching action with probability ϵ_t or exploits the best matching action with probability $1-\epsilon_t$, where ϵ_t is decreasing in t.

Contextual zooming (CZ) [9]: This algorithm adaptively creates balls over the joint action and context space, calculates an index for each ball based on the history of selections of that ball, and at each time step selects a matching action according to the ball with the highest index that contains the current context.

C. Yahoo! Today Module Simulations

In YTM each instance (user) has two contexts $(x_1, x_2) \in [0,1]^2$. We simulate the algorithms in Section VII-B for three different context sets in which the learning algorithms only decide based on (i) the first context x_1, (ii) the second context x_2, and (iii) both contexts (x_1, x_2) of the users. The m_T parameter of DISCOM for these simulations is set to the optimal value found in Theorem 1 (for $\gamma = 1$) which is $\lceil T^{1/4} \rceil$ for simulations with a single context and $\lceil T^{1/5} \rceil$ for simulations with both contexts. DISCOM is run for numerous z values ranging from $1/4$ to $1/2$. Table IV compares the performance of DISCOM, LinUCB, Hybrid-ϵ and CZ. All of the algorithms are evaluated at the parameter values in which they perform the best. As seen from the table the CTR for DISCOM with differential services is 16%, 5% and 7% higher

[9]The number of trainings and explorations required in the regret bounds are the worst-case numbers. In reality, good performance is achieved with a much smaller number of trainings and explorations.

Context	DISCOM	DISCOM (diff. serv.)	LinUCB	Hybrid-ϵ	CZ
x_1	6.37	7.30	6.31	5.92	4.29
x_2	6.14	6.45	4.72	6.14	4.39
(x_1, x_2)	5.93	6.61	5.65	6.15	4.24

TABLE IV
COMPARISON OF THE CTR$\times 10^2$ ACHIEVED BY CA 1 FOR DISCOM AND OTHER LEARNING ALGORITHMS FOR YTM.

z	1/4	1/3	1/2
CTR$\times 10^2$	5.13	5.29	6.14
CTR$\times 10^2$ in exploitations	5.14	5.34	6.45
Exploit %	98.9	97.7	90.2
Explore %	0.5	0.6	1.9
Train %	0.7	1.7	7.9

TABLE V
THE CTR, TRAINING, EXPLORATION AND EXPLOITATION PERCENTAGES OF CA 1 USING DISCOM WITH CONTEXT x_2 FOR YTM.

than the best of LinUCB, Hybrid-ϵ and CZ for contexts x_1, x_2 and (x_1, x_2), respectively.

Table V compares the performance of DISCOM, the percentage of training, exploration and exploitation phases for different control functions (different z parameters) for simulations with context x_2. As expected, the percentage of trainings and explorations increase with the control function. As z increases matching actions are explored with a higher accuracy, and hence the average exploitation reward (CTR) increases.

D. Music Dataset Simulations

Table VI compares the performance of DISCOM, LinUCB, Hybrid-ϵ and CZ for the music dataset. The parameter values used for DISCOM for the result in Table VI are $z = 1/8$ and $m_T = 4$. From the results is is observed that DISCOM achieves 10% improvement over LinUCB, 5% improvement over Hybrid-ϵ, and 28% improvement over CZ in terms of the average number of likes achieved for the users of CA 1. Moreover, the average number of likes received by DISCOM for the high type users (differential services) is even higher, which is 13%, 8% and 32% higher than LinUCB, HE and CZ, respectively.

E. Yahoo! Today Module with Drift Simulations

Table VII compares the performance of DISCOM-W with half window length ($\tau_h = 2500$) and $m_T = 10$, DISCOM (with m_T set equal to $\lceil T^{1/4} \rceil$ simulations with a single context dimension and $\lceil T^{1/5} \rceil$ for the simulation with two context dimensions), LinUCB, Hybrid-ϵ and CZ. For the results in the table, the z parameter value of DISCOM and DISCOM-W are set to the z value in which they achieve the highest number of clicks. Similarly, LinUCB, Hybrid-ϵ and CZ are also evaluated at their best parameter values. The results show the performance of DISCOM and DISCOM-W for differential services. DISCOM-W performs the best in this dataset in terms of the average number of clicks, with about 23%, 11.3% and 51.6% improvement over the best of LinUCB, Hybrid-ϵ and CZ, for types of contexts x_1, x_2, and (x_1, x_2), respectively.

Algorithm	DISCOM	DISCOM (diff. serv.)	LinUCB	Hybrid-ϵ	CZ
Avg. num. of likes	0.717	0.736	0.652	0.683	0.559

TABLE VI
COMPARISON AMONG DISCOM AND OTHER LEARNING ALGORITHMS FOR MD.

Contexts used	DISCOM-W (diff. serv.)	DISCOM (diff. serv.)	LinUCB	Hybrid-ϵ	CZ
x_1	6.3	5.5	5.1	3.0	2.4
x_2	5.1	4.2	4.2	4.6	2.4
(x_1, x_2)	6.9	3.8	4.6	4.1	2.3

TABLE VII
THE CTR$\times 10^2$ OF DISCOM-W AND DISCOM FOR DIFFERENTIAL SERVICES, AND THE CTR OF OTHER LEARNING ALGORITHMS FOR YTMD.

VIII. CONCLUSION

In this paper we considered novel online learning algorithms for content matching by a distributed set of CAs. We have characterized the relation between the user and content characteristics in terms of a relevance score, and proposed online learning algorithms that learns to match each user with the content with the highest relevance score. When the user and content characteristics are static, the best matching between content and each type of user can be learned perfectly, i.e., the average regret due to suboptimal matching goes to zero. When the user and content characteristics are dynamic, depending on the rate of the change, an approximately optimal matching between content and each user type can be learned. In addition to our theoretical results, we have validated the concept of distributed content matching on real-world datasets. An interesting future research direction is to investigate the interaction between different CAs when they compete for the same pool of users. Should a CA send a content that has a high chance of being liked by another CA's user to increase its immediate reward, or should it send a content that has a high chance of being disliked by the other CA's user to divert that user from using that CA and switch to it instead.

REFERENCES

[1] S. Ren and M. van der Schaar, "Pricing and investment for online TV content platforms," *IEEE Trans. Multimedia*, vol. 14, no. 6, pp. 1566–1578, 2012.

[2] S. Song, H. Moustafa, and H. Afifi, "Advanced IPTV services personalization through context-aware content recommendation," *IEEE Trans. Multimedia*, vol. 14, no. 6, pp. 1528–1537, 2012.

[3] C. Xu, J. Wang, H. Lu, and Y. Zhang, "A novel framework for semantic annotation and personalized retrieval of sports video," *IEEE Trans. Multimedia*, vol. 10, no. 3, pp. 421–436, 2008.

[4] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proc. of the 19th International Conference on World Wide Web*. ACM, 2010, pp. 661–670.

[5] M. Saxena, U. Sharan, and S. Fahmy, "Analyzing video services in web 2.0: a global perspective," in *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2008, pp. 39–44.

[6] R. Mohan, J. R. Smith, and C.-S. Li, "Adapting multimedia internet content for universal access," *IEEE Trans. Multimedia*, vol. 1, no. 1, pp. 104–114, 1999.

[7] M. Schranz, S. Dustdar, and C. Platzer, "Building an integrated pan-European news distribution network," in *Collaborative Networks and Their Breeding Environments*. Springer US, 2005, pp. 587–596.

[8] A. Boutet, K. Kloudas, and A.-M. Kermarrec, "FStream: a decentralized and social music streamer," in *Proc. of the International Conference on Networked Systems (NETYS)*, pp. 253–257.

[9] A. Slivkins, "Contextual bandits with similarity information," in *Proc. of the 24th Annual Conference on Learning Theory (COLT)*, vol. 19, June 2011, pp. 679–702.

[10] T. Lu, D. Pál, and M. Pál, "Contextual multi-armed bandits," in *Proc. of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 9, May 2010, pp. 485–492.

[11] Y. Deshpande and A. Montanari, "Linear bandits in high dimension and recommendation systems," in *Proc. of the 50th Annual Allerton Conference on Communication, Control, and Computing*, 2012, pp. 1750–1754.

[12] P. Kohli, M. Salek, and G. Stoddard, "A fast bandit algorithm for recommendations to users with heterogeneous tastes," in *Proc. of the 27th Conference on Artificial Intelligence (AAAI)*, July 2013, pp. 1135–1141.

[13] N. Sahoo, P. V. Singh, and T. Mukhopadhyay, "A hidden Markov model for collaborative filtering," *MIS Quarterly*, vol. 36, no. 4, pp. 1329–1356, 2012.

[14] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *Internet Comput.*, vol. 7, no. 1, pp. 76–80, 2003.

[15] K. Miyahara and M. J. Pazzani, "Collaborative filtering with the simple Bayesian classifier," in *PRICAI 2000 Topics in Artificial Intelligence*. Springer, 2000, pp. 679–689.

[16] S. D. Roy, T. Mei, W. Zeng, and S. Li, "Empowering cross-domain internet media with real-time topic learning from social streams," in *Multimedia and Expo (ICME), 2012 IEEE International Conference on*. IEEE, 2012, pp. 49–54.

[17] S. Roy, T. Mei, W. Zeng, and S. Li, "Towards cross-domain learning for social video popularity prediction," *IEEE Trans. Multimedia*, vol. 15, no. 6, pp. 1255–1267, Oct 2013.

[18] D. Bouneffouf, A. Bouzeghoub, and A. L. Gançarski, "Hybrid-ε-greedy for mobile context-aware recommender system," in *Advances in Knowledge Discovery and Data Mining*. Springer, 2012, pp. 468–479.

[19] E. Hazan and N. Megiddo, "Online learning with prior knowledge," in *Learning Theory*. Springer, 2007, pp. 499–513.

[20] M. Dudik, D. Hsu, S. Kale, N. Karampatziakis, J. Langford, L. Reyzin, and T. Zhang, "Efficient optimal learning for contextual bandits," *arXiv preprint arXiv:1106.2369*, 2011.

[21] W. Chu, L. Li, L. Reyzin, and R. E. Schapire, "Contextual bandits with linear payoff functions," in *Proc. of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 15, April 2011, pp. 208–214.

[22] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, p. 235256, 2002.

[23] M. Naaman, "Social multimedia: highlighting opportunities for search and mining of multimedia data in social media applications," *Multimedia Tools and Applications*, vol. 56, no. 1, pp. 9–34, 2012.

[24] L. L. Minku, A. P. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 5, pp. 730–742, 2010.

[25] A. Narasimhamurthy and L. I. Kuncheva, "A framework for generating data to simulate changing environments," in *Proc. of the 25th IASTED International Multi-Conference: Artificial Intelligence and Applications*, February 2007, pp. 384–389.

[26] M. van der Schaar, J. Xu, and W. Zame, "Efficient online exchange via fiat money," *Economic Theory*, vol. 54, no. 2, pp. 211–248, 2013.

[27] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Consulted*, vol. 1, no. 2012, p. 28, 2008.

[28] J. Langford and T. Zhang, "The epoch-greedy algorithm for contextual multi-armed bandits," in *Proc. of the 20th International Conference on Neural Information Processing Systems (NIPS)*, vol. 20, 2007, pp. 1096–1103.

[29] C. Tekin and M. van der Schaar, "Contextual online learning for multimedia content aggregation," *arXiv:abs/1502.02125*, 2015.

[30] J. Gao, W. Fan, and J. Han, "On appropriate assumptions to mine data streams: Analysis and practice," in *Proc. of the 7th IEEE International Conference on Data Mining (ICDM)*, 2007, pp. 143–152.

[31] M. M. Masud, J. Gao, L. Khan, J. Han, and B. Thuraisingham, "Integrating novel class detection with classification for concept-drifting data streams," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2009, pp. 79–94.

Cem Tekin Cem Tekin is an Assistant Professor in Electrical and Electronics Engineering Department at Bilkent University, Turkey. From February 2013 to January 2015, he was a Postdoctoral Scholar at University of California, Los Angeles. He received the B.Sc. degree in electrical and electronics engineering from the Middle East Technical University, Ankara, Turkey, in 2008, the M.S.E. degree in electrical engineering: systems, M.S. degree in mathematics, Ph.D. degree in electrical engineering: systems from the University of Michigan, Ann Arbor, in 2010, 2011 and 2013, respectively. His research interests include machine learning, multi-armed bandit problems, data mining, multi-agent systems and game theory. He received the University of Michigan Electrical Engineering Departmental Fellowship in 2008, and the Fred W. Ellersick award for the best paper in MILCOM 2009.

Mihaela van der Schaar Mihaela van der Schaar is Chancellor Professor of Electrical Engineering at University of California, Los Angeles. Her research interests include network economics and game theory, online learning, dynamic multi-user networking and communication, multimedia processing and systems, real-time stream mining. She is an IEEE Fellow, a Distinguished Lecturer of the Communications Society for 2011-2012, the Editor in Chief of IEEE Transactions on Multimedia and a member of the Editorial Board of the IEEE Journal on Selected Topics in Signal Processing. She received an NSF CAREER Award (2004), the Best Paper Award from IEEE Transactions on Circuits and Systems for Video Technology (2005), the Okawa Foundation Award (2006), the IBM Faculty Award (2005, 2007, 2008), the Most Cited Paper Award from EURASIP: Image Communications Journal (2006), the Gamenets Conference Best Paper Award (2011) and the 2011 IEEE Circuits and Systems Society Darlington Award Best Paper Award. She received three ISO awards for her contributions to the MPEG video compression and streaming international standardization activities, and holds 33 granted US patents.