IEEE 23rd International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems

CPSys: A system for mobile video prefetching

Ali Gouta*, David Hausheer†, Anne-Marie Kermarrec‡,
Christian Koch†, Yannick Lelouedec*, Julius Rückert†
*Orange Labs,†TU Darmstadt, ‡Inria
Email: *ali.gouta@orange.fr, †hausheer@ps.tu-darmstadt.de, ‡anne-marie.kermarrec@inria.fr, †ckoch@ps.tu-darmstadt.de,
*yannick.lelouedec@orange.com, †rueckert@ps.tu-darmstadt.de

***Abstract*—Online media services are reshaping the way video content is watched. People with similar interests tend to request same content. This provides enormous potential to predict which content users are interested in. Besides, mobile devices are commonly used to watch videos which popularity is largely driven by its social success. In this paper, we design CPSys a Central Predictor System to prefetch relevant videos for each user. To fine tune our prefetching system, we rely on a large dataset collected from a large mobile carrier in Europe. The rationale of our prefetching strategy is first to form a graph and build implicit or explicit ties between similar users. On top of this graph, we propose the Most Popular and Most Recent (MPMR) policy to predict relevant videos for each user. We show that CPSys can achieve high performance with respect to the correct prediction ratio and by significantly reducing the traffic overhead. We further show that CPSys outperforms other prefetching schemes that have been presented and studied in the state of the art. At the end, we provide a proof-of-concept implementation of our prefetching system.**

I. Introduction

Today, mobile devices are commonly used to watch videos everywhere. Within a few years mobile devices are likely to become the users' preferred choice for accessing the Internet [1] while according to [1] multimedia content represents already a significant portion of the mobile traffic today. This growing trend is to a large extent driven by social networks. Online social networks (OSNs) are reshaping the way videos are being consumed. That is by boosting popularity of video content within groups of users with similar interest [2] and by providing viewing recommendation for each user. EdgeRank [3] is used by Facebook to sort items on the news feed of the individual users based on affinity, weight and time decay scores. These key factors drive users to conduct a particular behavior when browsing their news feed and allow for a prediction of content a user is interested in. This social or interest based interaction can be leveraged by networking actors, in particular over-the-top (OTT) content providers and content delivery networks (CDNs), to predict future behavior of users and decide if it is worth pushing videos to the interested users at a particular time. Hence, if properly designed, prefetching videos can alleviate the network during peak traffic periods, i.e. flash crowds. Besides, it can improve the user experience since it avoids buffering delays or stalling of the streaming video as the content can be played from local storage.

[1] http://www.morganstanley.com/about/press/articles/4659e2f5-ea51-11de-aec2-33992aa82cc2.html

In this paper, we design and implement CPSys, a Central Predictor system to prefetch videos on users' mobile devices. Our prefetching scheme aims to answer the 3 following questions:

- *Which content should be prefetched?* To determine which content the user is interested in is hard in general. The lifetime video views combined with the preferences of the users are key factors to build an accurate prediction model. CPSys assumes that the user runs an application on his device which reports information to a central server which holds all users' profiles and their past activities. This central server builds and maintains a similarity graph. This graph is either inferred from the user's social ties or built based on collaborative filtering techniques. We name the former a social graph and we name the latter an interest graph. When CPSys runs in social mode, the application running on the user device reports to the central server the social ties of that particular user, hence we build and maintain the social graph based on these reports. When CPSys runs in interest mode, the graph is built and maintained as the following: Everyday, we compute the affinity scores between users and reassign the edges with respect to these new scores.
  Finally, on top of this graph, we identify and maintain for each user a list of relevent videos that would interest him or her. We infer this list from the videos watched by the neighbors.
- *When to trigger prefetching?* We define two control mechanisms: a network-oriented and a state-transition control mechanism. The former allows an efficient use of network resources while the latter aims at controlling the prefetcher agent running on the user device. Combined, these control mechanisms do not allow the agent to prefetch videos aggressively or randomly.
- *How many videos are to be prefetched?* This is a design choice. We differentiate between 2 kinds of users: Heavy and light users. We correlate the number of videos to prefetch with the user's past activity and conclude on the number of videos to be prefetched.

Our system design can be leveraged by all networking actors, in particular by OTTs, CDN providers, and telcos. We implemented several features to make it open and flexible for future extension.

The rest of the paper is organized as follows. Section II

Dataset name	Number of unique users	Number of unique videos	Number of requestes	Service provider	Typical URI
YT (Youtube dataset)	3,179,296	10,676,156	64,722,755	Google CDN	r8—sn-4g57kue6.youtube.com/videoplayback?id=↩ **d1875abcee9b6d33**&itag=36&source=youtube&....
FB (Facebook dataset)	399,645	2,856,321	14,305,404	Akamai	video.ak.fbcdn.net/hvideo-ak-prn2/v/↩ **1608327_750958311600439_982850231**_n.mp4?...

TABLE I. PROPERTIES OF THE TWO USED DATASETS

describes the related works. Section III represents observations on the behavior of clients through analyzing the used traffic trace. In Section IV, we introduce our system and configure it with respect to the observations drawn from the previous section. In Section V, we extensively evaluate our system with different assumptions. Section VI, we provide a proof-of-concept implementation of our system. Finally, we conclude the paper in Section VII.

II. RELATED WORKS

Watching videos with mobile devices becomes more and more popular. In [4] and [5], authors deeply investigated the mobile users' viewing behavior when the mobile devices are connected to 3G and WiFi networks. Their study focuses mainly on TV programs watched on mobile devices, while in this paper we provide insights into video properties published in two widely user generated content (UGC) services, namely Facebook and YouTube UGCs. Interestingly, we observe a common behavioral pattern between our analysis and those provided in [5] regarding the evolution of video popularity. Our analysis on Youtube videos -except music category- shows a sharp increase in popularity immediately after uploading the video. This increase is followed by a fast decrease in the number of views across time. This same pattern is observed for TV programs watched on mobile devices.

Recent studies have brought forth the benefits of prefetching videos on mobile devices. NetTube [6] and SocialTube [7] were designed to leverage P2P overlays to download YouTube videos. In these systems, authors proposed a prefetching scheme for prefetching prefixes of videos to improve the user experience at the joining phase. In SocialTube, the authors assumed optimistic hypothesis to evaluate their prefetching strategy. They limited their study to 2000 videos shared among 5000 nodes. While this holds reasonable to assess the maximum performance the system may achieve, we believe that this introduces a bias on the performance results. In contrast, the present paper exploits real traffic traces collected over a large mobile carrier in Europe; thus all the specific characteristics of mobile video traffic are captured and taken into account in the assessment works reported in this paper.

In [8], authors show that prefetching has potential benefits regarding energy savings. Prefetching when the device is connected to Wifi can reduce the energy consumption by 10% with respect to a 3G connection. Yet, the authors did not investigate the fundamental and prior question that should be addressed: which content should be prefetched to individual users? Even if connected to Wifi, an aggressive prefetching strategy, i.e. prefetching all videos, would drain the battery very fast which, as well, leads to a bad user experience. In this respect, we focus on the primarily question: *what to prefetch ?* Once we identify the video candidates, we address the second question: *when to prefetch?*

Mohan et al. [9], proposed to prefetch advertisements (ads), to achieve energy savings. In this paper we do not limit our study to ads. Instead we prefetch all videos that may interest an individual user. Finamore et al. [10] proposed to periodically prefetch bundles of popular content videos on mobile devices. In the prefetching context, we believe that the term *popularity* has no absolute meaning. A content might be locally popular inside a group of users sharing similar interests, but not globally popular and vice versa.

As briefly discussed above, none of the presented related works address all fundamental questions that we see as being of fundamental importance to prefetching. To this end, we carry out analysis and draw lessons using real mobile traffic traces. Based on these observations, we design CPSys a network-friendly prefetching system. Then, we assess the proposed mechanism using real traffic traces. At the end, we provide a proof of concept implementation of CPSys.

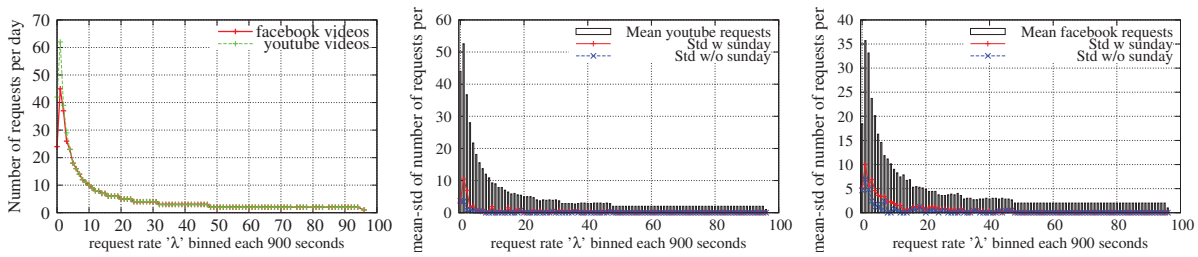
III. TRAFFIC ANALYSIS

In this section, we introduce our dataset and provide analysis and findings on users' behavior. Then, we use these findings to infer the design principles of our prefetching system.

A. Dataset

We rely on a large dataset gathered at all G_i interfaces of all Gateway GPRS Support Nodes (GGSN) deployed by a major mobile carrier in France. The dataset consists of logs of video streaming sessions generated by all connected devices of the carrier's subscribers. The logs were collected from 8 January 2014 to 28 April 2014. Due to maintenance reasons, the monitoring infrastructure was disabled for 27 days, which makes the real period of data collection lasting about 94 days. These disruptions do not introduce any bias in the data analysis and simulations reported in this paper since they are not achieved over the whole duration of the dataset. We limit our study to 2 subsets of video traffic: requests for YouTube (YT) videos and requests for Facebook (FB) videos. Table I gives an overview of both these subsets of traffic. Parsing the HTTP header in FB and YT traffic flows enables to extract the unique video identifier (noted *reference ID*) requested by the users. For illustrative purpose typical URIs from YT and FB are given in Table I, the field in bold pointing to this *reference ID* of the video. To preserve confidentiality and privacy, our dataset is anonymized during an early stage in the collection process. When capturing the log, the user confidential information are pre-processed and changed into a unique identifier, hence we use a fully anonymized dataset to conduct our analysis.

In the following, we provide empirical traffic observations and Findings (noted **F**), upon which we establish the design principles of our prefetching system. Due to space limitation, we intentionally omit to show the figures carried by the two



(a) YT+FB data (b) YT data (c) FB data

Fig. 1. a: Relationship between the number of requests per day and the daily request inter-arrival time on 01/13/2014; b and c : on the week starting from 01/13/2014

first findings. Besides, the two first findings have been largely reported in several previous studies, yet they are still important for the design of our prefetching system.

First, we observed that video popularity follows Zipf distribution for both YT and FB videos, i.e. most of the videos are requested few times while few cumulates most of the views.

F1: Regarding prefetching, popularity remains an important factor to maximize the prediction accuracy and to best manage network resource utilization: a safe prefetching strategy would be to favor the prefetching of popular videos.

Second, we observed that few users request far more frequently video contents – we call them heavy users – while the majority is less active – we call them light users. Now with respect to prefetching, predicting the behavior of light users is hard in general and turns into a typical cold start situation where it is hard to learn the preferences of the user from a small set of viewed videos.

F2: It is important to adapt the prefetching strategy with regard to the user activity. Aggressively prefetching at light users' videos does not make sense: Our prefetching system differentiates between heavy and light users based on their past activity.

In the rest of this section we investigate 3 more traffic properties:*relationship between number of requested videos per day and request inter-arrival rate*, *video lifetime distribution*, and *load variation across the day*.

B. Relationship between number of requested videos per day and request inter-arrival rate

Figure 1(a) quantifies the relationship between the number of requests and the request inter-arrival time of the YT and FB videos during one given day; here Monday, January 13, 2014.

Given a row data - a vector of $(x_\lambda, y_N)_u$ values representing one single user, where x_λ represents the inter-arrival time of requests over one day and y_N represents the number of viewed videos per day - on the x-axis, we start creating bins of 900 second long. This subsequently generates 96 bins to cover all the day. Then each user is associated to one bin. The bin 0 in the x-axis refers to users having their inter-arrival request time ranging from 0 to 900 seconds. Bin 1 corresponds to the range [900 seconds, 2*900 seconds[, etc. On the y-axis, for each of these 96 groups, we show the number of requests per day (y_N) of the 99-percentile most active user within each bin. Figure 1(a) shows that the activity of users could be modeled and quantified with an exponential decay. Users do not request more than 3 videos per day when their request inter-arrival time is higher than $20 * 900$ seconds.

Figures 1(b) and 1(c) generalize this observation for the rest of the days of the week (until 20 January). In these figures, users are binned on the x-axis as per the daily average inter-arrival time of the requests they generated in the week of Monday, January, 13, 2014. On the y-axis, for each of the 96 bins, we show the mean over the seven days of number of requests of the 99-percentile most active user within each bin. The standard deviation is also given twice: once over the seven days, and once over six days from Monday to Saturday (excluding Sunday).

Figures 1(b) and 1(c) show that the request inter-arrival time remains slightly similar across the days of the week. The standard deviation gets quickly close to zero for the least active users and it also remains relatively low for the heaviest users. The standard deviation is slightly higher when it includes Sunday. This illustrates that users have more heterogeneous consumption behaviors on Sunday than the other days. On Sunday, we record a lower activity on mobile devices. This suggests that the majority of users consume less FB and YT videos on their mobile devices while a minority is much more active on Sundays. The patterns are quite similar for YT (Figure 1(b)) and FB (Figure 1(c)); only the mean request inter-arrival time of the heaviest users is higher for YT.

F3: The request inter-arrival time might be used to identify the heaviest users from the least active ones in order to enforce them a specific prefetching strategy. Moreover Figures 1(b) and 1(c) provide additional insights to fine tune the prefetching system: The 99-percentile most active users request no more than 35 FB videos and 52 YT videos per day on average, which gives an insight on the daily number of videos to prefetch.

C. Video lifetime distribution

In Figure 2, we show the lifetime distribution of YouTube videos across the time. We limited our study to videos that have been uploaded to YouTube on 8 January 2014, which is the starting day of the dataset. We also excluded music video category since this category exhibit a different popularity dynamics with respect to the rest of categories. Popularity of music videos is more likely to sustain across time [11], while

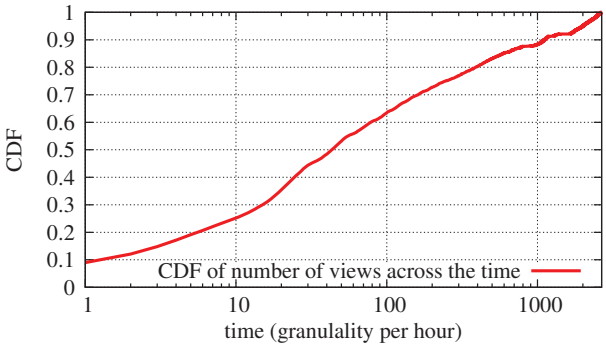


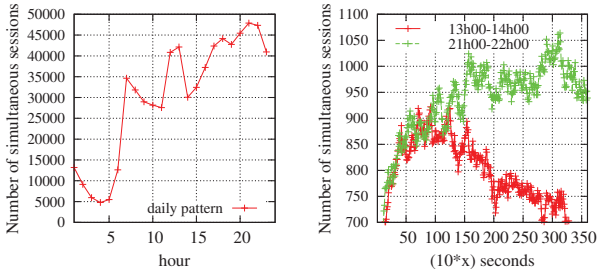
Fig. 2. Lifetime distribution of videos made available on January 8 2014

-as we show in this part- other categories do not necessarily stick to this finding.

In figure 2, we align all videos to the same starting point and we plot the cumulative distribution of the number of views of these videos. The figure clearly shows that most of the views happen in a short time frame after the videos are made available: 10% the first hour and 40% the first day.

F4: According to Cha et al. [12] a large part of content items is immutable which means that users tend to lose interest in an item immediately after they consumed it. Figure 2 confirms that any prefetching strategy shall be proactive and quickly anticipate the interest of each user towards each video.

D. Load variation across the day



(a) Load variation during the day (b) Load variation during peak hours

Fig. 3. (a) Number of active sessions across the day; (b) Zoom in during peak hours

We plot in Figure 3(a) the aggregated number of simultaneous YT and FB sessions in the dataset across one representative day. As expected, it follows a classic daily pattern with peaks at 1pm and in the evening. Figure 3(b) concentrates during the most loaded hours: 1-2pm and 9-10pm. At a granularity of seconds we observe that traffic is not uniformly distributed. A clever prefetching strategy would be to leverage the local minima in these most loaded hours to push contents on mobile devices. Digging further, one might investigate the best way to allocate the mobile spectrum. One possible solution would be to efficiently reuse white spaces [13] and push contents during these periods. Yet, we leave this for future work.

F5: Load is not uniformly distributed, including at peak hours. Therefore, prefetching should be scheduled either at off-peak hours or at much smaller time scales at the least loaded instants during the peak hours, and in coordination with the mobile carrier's resource allocation scheme.

IV. System Design

CPSys (Central Predictor System) is designed to leverage the user's interest or social ties to determine a personalized list of content items to be prefetched for each user. Figure 4 depicts the CPSys architecture. The system consists of two main components:

- *Prefetcher agent*. Installed on the user device this component runs as a background service to ensure two main functions. First it provides the centralized predictor with reports on the user' activities. Second it triggers and controls the prefetching of the videos from a list it receives from the centralized predictor.
- *Central predictor (CP)*. This component holds and updates profiles of all users running the prefetcher agents based on the reports sent by these agents. Finally it exploits all these sets of information to predict the candidate videos to prefetch for each user running the prefetcher agent.

The 3 following questions are used to drive the design of our system: *What to prefetch? When to prefetch? How many videos to prefetch?*

A. What to prefetch?

Our strategy is first to identify users with common interests. The rationale is to affiliate for each user a group of users - we call them neighbors - who tend to request similar content items, hence we build a directed graph. Second, on top of this graph and for a given user X, the centralized predictor tracks the videos that has been requested by neighbors of user X and defines a personalized list of prefetching candidates. In the following, we detail how the CP creates the graph and updates the users' profiles:

1) Building the graph: The graph component (B1 on Figure 4) is one of the most important components of the CP. It builds and maintains ties between users based on social or interest affinity. It implements the following two interfaces:

a) Social graph interface (SG): The prefetcher agent, installed on the user device reports the IDs of all social neighbors running the prefetcher agent to the central predictor. Hence, the social graph is inferred from OSNs like Facebook or Google+. In our proof-of-concept, we used the Facebook API to implement this feature and make the central predictor aware of the user's social ties.

b) Interest graph interface (IG): The central predictor updates in daily routine the list of neighbors affiliated to each user. It computes similarities between users based on all past and recorded user preferences, hence re-affecting the implicit ties with respect to the new similarity scores. The more we learn about the user preferences, the more accurate

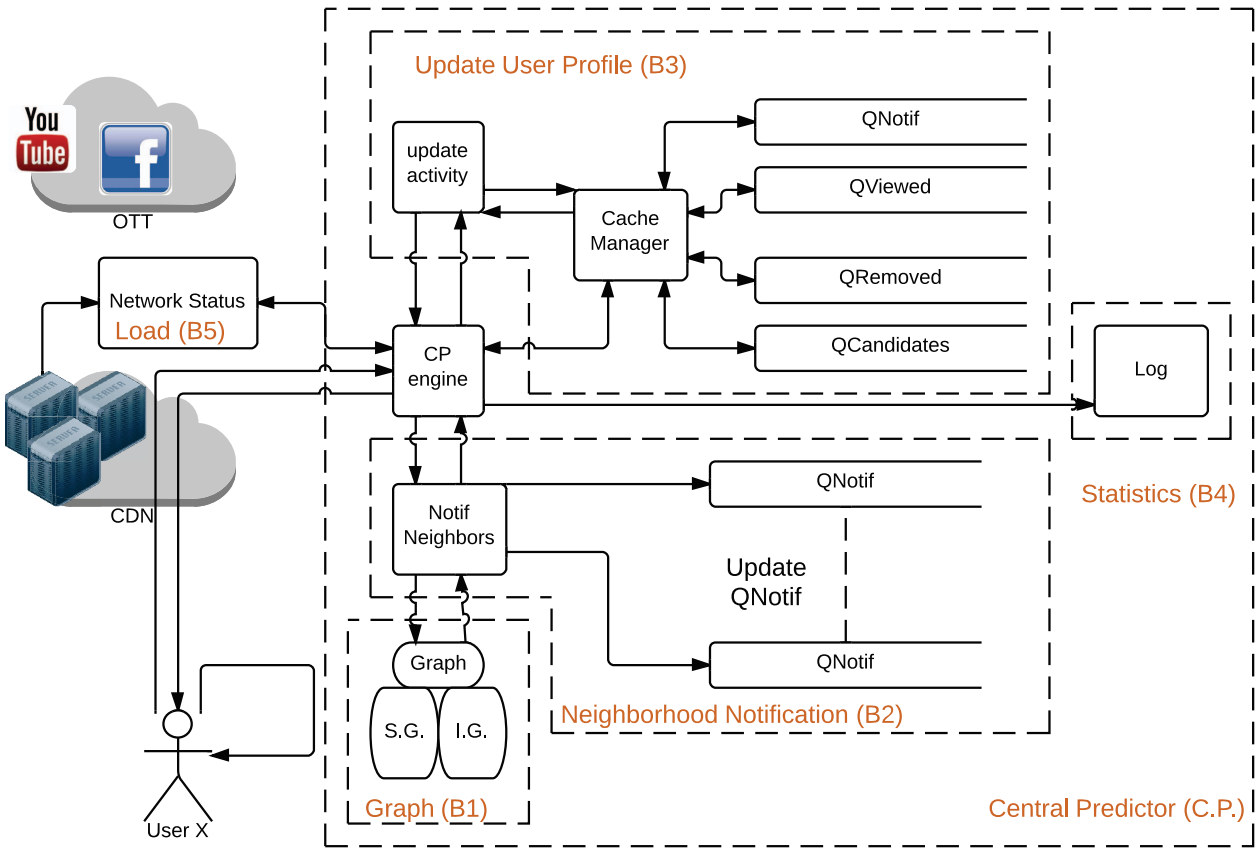


Fig. 4. CPSys design

the prediction model can become. We use the Jaccard index [14] to compute the Affinity (A) score between all users:

$$A(u,y)_d = \frac{|L_u(v)_{t..d-t-1} \cap L_y(v)_{t..d-t-1}|}{|L_u(v)_{t..d-t-1} \cup L_y(v)_{t..d-t-1}|} \quad (1)$$

$L_u(v)_{t..t-d}$ is the set of videos viewed by user u over the time window of d days. We associate for each user the K-Nearest Neighbors (KNN), i.e. the ones with the highest affinity scores. The decision for an appropriate K value is a design choice.

2) Content selection process: Having established the graph, when user X requests a content item v at instant t, the centralized predictor captures this request in real time and proceeds as follows:

- Neighborhood Notification (B2 on Figure 4). The CP manages (creates and updates) a queue named *QNotif* which contains the identifiers, i.e. URLs, of the videos watched by the user's neighbors. We call it QNotif because it is updated only if one of the user's neighbors watches a video. More precisely, each element in QNotif -named index- is a data-structure composed of the unique identifier of a video and several attributes of the video, including its popularity, the source(s) of the request(s) for that video (the neighbor(s) who requested the same content), its freshness (the date of the latest request for that content), and the pointers to the next and previous indexes in QNotif. QNotif may implement different policies to rank the indexes in the queue. We opted for the Most Popular Most Recent (MPMR) policy for the queue QNotif in CPsys, as detailed below.

- Update User Profile (B3 on Figure 4): The CP updates the profile of user X. First, it updates the request rate associated to this user and increments the number of requested videos viewed per day. Then the CP inserts the index of the viewed video into the Qviewed queue. In CPsys we do not prefetch the same content multiple times. The rationale behind this design choice is twofold:
First, users watch videos at most once [12]. This is especially the case for, e.g. catch-up [15] or user-generated content, where users tend to watch the content only one time.
Second, we consider the local cache of a user being large enough to hold content items for a considerable period of time before removing it. As a result, requests for already prefetched items can be easily served

locally, even for multiple requests.

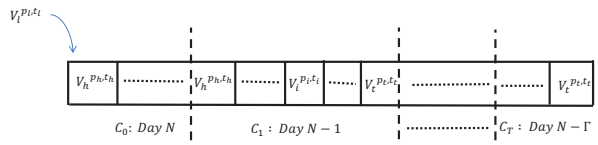


Fig. 5. QNotif: Data structure which holds the prefetching candidates

Figure 5 presents the queue QNotif implemented with the MPMR policy. QNotif is divided into a set of classes $(C_{j=0..\Gamma})$:

- Each class C_j includes at least the indexes of the videos that have been watched by a part or all of the user's neighbors on day $d = (N - j)$, where N points to the actual day (d equals to N in that case).
- Γ is a parameter, expressed in terms of number of days. Γ is used to prevent the queue from growing indefinitely. The setting of this parameter is a design choice. The higher the value of Γ, the more costly the look_up and update operations will be. Given that the number of views in the considered dataset drops significantly 3 days after the upload (c.f. Finding F4 in Section III-C), we set Γ to 3.

In MPMR and within each class C_j, the rank of the index is attributed with respect to the popularity (p) as a first criteria, then with respect to the freshness of the content. The index *heading* the class is referred by $(v_h{}^{p_h,t_h})$ which should point to the most viewed content by neighbors, then as far as we iterate through the list of indexes, the popularity should either remain the same or drops until we reach the *tail* of the class $(v_t{}^{p_t,t_t})$.

Based on the notations used in Figure 5, we illustrate how QNotif is maintained with respect to the MPMR policy, we suppose that one of the neighbors of user X requests a content item V_l. As a result, CP updates the list of indexes of the user X's QNotif. Algorithm 1 and the following paragraphs detail where exactly in the list to insert the index v_l, which is a pointer to content item V_l.

Require: $Q_n(v), v_l$;
1: **if** $v_l \notin Q_n(v)$ **then**
2: insert(v_l, N, $p_{i_{target}} > 1$, $p_{i_{target}} == 1$)
3: **else**
4: $d_l \leftarrow get\_day(Q_n(v_l))$
5: $p_l \leftarrow get\_popularity(Q_n(v_l))$
6: $p_l \leftarrow p_l + 1$
7: **if** $d_l == N$ **then**
8: move(v_l, N, $p_{i_{target}} + 1 > p_l$, $p_l \geq p_{i_{target}}$)
9: **else**
10: $d_l \leftarrow d_l + 1$
11: move(v_l, d_l, $p_{i_{target}} > p_l$, $p_l \geq p_{i_{target}}$)
12: **end if**
13: **end if**

Algorithm 1: Update of user X's QNotif $(Q_n(v))$ upon the request for video V_l with index v_l by one of user X's neighbors

- *Line 1,2*: This is the case where index v_l does not exist in $Q_n(v)$, hence the index is created and inserted into class C_0. The parameter p refers to the number of views per video v. A newly created index starts always with $p = 1$, since only one of the neighbors watched the content. When we insert v_l, first, we determine its target location which is index $v_{i_{target}}$ where : $(p_{i_{target}-1} > 1)$ and $(p_{i_{target}} = 1)$, then we shift each index after this target location $v_{i_{target}}$ by one position to make room for the new insertion.

Now, if the index v_l already exists within QNotif, we take both the C_j and p parameters to decide where to move it.

- *Line 7*: This line corresponds to the case where C_l equals to C_N. This index remains within class N and only the number of views (p_l) is incremented. Then, v_l is moved ahead to the position before index $v_{i_{target}}$ with property $(p_{i_{target}-1} > p_l)$ and $(p_{i_{target}} <= p_l)$.
- *Line 8*: This is the case where $C_{l<N}$. In this case, v_l jumps to class C_{l+1}, increments the (p_l) score, and is moved ahead before index i_{target} with property $(p_{i_{target}-1} > p_l)$ and $(p_{i_{target}} <= p_l)$.

In the next section, we show that MPMR outperforms other policies by improving the prediction accuracy and decreasing the overhead. In particular we compare MPMR to LRU and FIFO policies.

B. When to prefetch?

To efficiently manage the prefetching process, the system includes a double control mechanism, consisting of a network-oriented and a state-transition control mechanism. The prefetching -triggered and performed by the user device- is hindered until both control mechanisms meet.

1) Network control mechanism: The prefetching shall be hindered when the network is overloaded. And, ideally, it should be achieved in coordination with the mobile carrier's resource allocation scheme (See Finding F5 in Section III-D).

Hence, the Network status component (B5 on Figure 4) is used to monitor and report the traffic load to the centralized predictor. If the load exceeds a certain threshold, prefetching is not allowed to be performed. This control mechanism is still an ongoing work, therefore we leave this part for future works.

2) State-transition control mechanism: The prefetcher agent is also controlled by a state-transition control mechanism. Figure 6 shows the transition states that the prefetcher agent should follow-up.

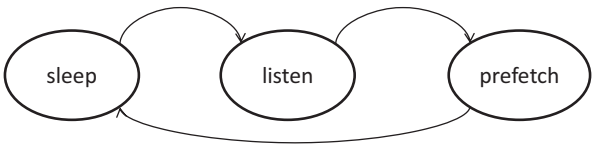


Fig. 6. Transition-state control mechanism running on the prefetcher agent

Figure 6 depicts the 3 states the prefetcher agent goes through. By default, the prefetcher agent is in the "listen" state and tracks all user requests. Then, if the network conditions are met; therefore, the agent switches to the "prefetch" state and

the video prefetching process is triggered. When this process ends, the agent turns to the "sleep" state for a certain period of time Δ, predefined at the central predictor. Then it turns back to the "listen" state.

This double-check control mechanism aims at maintaining a solid control on the prefetcher agent, so as to prevent too aggressive video prefetching plans especially for light users. This is in line with the Findings F2 and F3 from Sections III-A and III-B.

C. How many to prefetch?

Finding F3 in Section III-B. shows that a small set of users behaves as heavy users. However, the users may change their behavior across the time. Hence, the number of prefetched videos should also follow the evolution of each user's behavior. In CPsys, the Update User Profile block (B3 on Figure 4) models the user behavior as a function of time, based on his past activity, and it determines the final list of videos to prefetch with these three parameters: $[\tilde{S}_{max}]_d$, $N_{prefetch}$ and p_{th}.

- $[\tilde{S}_{max}]_d$ is a prediction of the number of content items the user would request at day d. It is updated on a daily basis and equals to the average number of requested videos per day over the 10 past days.

$$[\tilde{S}_{max}]_d = \frac{\sum_{i=d-11}^{d-1} \textit{(number of requests per day)}_i}{10} \tag{2}$$

- $N_{prefetch}$ is the maximum theoretical number of videos from the queue QNotif that the prefetcher agent is allowed to prefetch when the prefetching process is executed. r is a scale factor bounded between 0 and 1 and used to approxiamte the number of prefectched items on one user device and on one day to $\tilde{S}_{max}$. For example, if prefetching is executed 3 times on one user device during the day and r is equal to 0.34, then the number of items that will be prefetched during that day will be equal to $\tilde{S}_{max} + 3$.

$$N_{prefetch} = |\tilde{S}_{max} * r + 1| \tag{3}$$

- The threshold popularity score, p_{th}, is used to achieve a final filter among the $N_{prefetch}$ best ranked candidate videos in QNotif. The rational is to avoid prefetching very unpopular contents. Hence only the subset of the videos satisfying the property $(p_i >= p_{th})$ are moved to the QCandidates queue in the Update User profile block and communicated as a list to the prefetcher agent on the user device to be prefetched sequentially. In the next section, we investigate the impact of fine-tuning this parameter p_{th} on the performances of CPsys.

The queue QRemoved on Figure 4 is just used for debugging purposes. If the user requests a content which had already been prefetched previously but removed before the user watches it, due to storage capacity limitation for example, QRemoved is updated with the index of the removed video. Later, this information is used to feed the Statistics component (B4 on Figure 4).

V. Trace-driven simulation experiments

We developed Prefsim [2], written in Java, a simulator implementing the CPSys architecture as presented in Figure 4 and described in the previous Section. Prefsim runs either in social or interest mode. It is a trace-driven simulator. In both modes, Prefsim requires a traffic trace as input file which contains the timestamp, the userID, the videoID, and the duration for individual user sessions. Additionally, if it runs in social mode, the simulator requires the social graph as input file.

The task performed by the prefetching system might be considered as an instance of a recommendation problem: the system should be able to predict the user's interest and to timely prefetch the content of interest. The approaches applied in studies in the area of recommendations usually rely on the collection and exploitation of preferences expressed by users - mostly ratings - to build recommendation algorithms and engines, as well as to assess their performances - mostly recall and precision - [16].

In recommendation systems, it is commonly known that a very small proportion of users express their opinions on items, instead we rely on real requests and not users' expressions. The Finding F1 in Section III-A suggests that the heavy long tail of the video popularity distribution leads to a high sparsity levels of the user-video matrix, which includes all users and videos from the trace and describes how the two are linked by observed sessions. Even if the prediction runs perfectly, it is expected that the recall -later we call it Hit-Ratio (HR)- of approaches based on this matrix will be too low to show a considerable benefit in the context of prefetching.

We evaluate CPsys based on 3 metrics which are network oriented. We evaluate the following:

- Correct Prediction Ratio (CPR): Ratio of requests served from the user's local cache out of the number of prefetched videos. CPR refers to the precision of our prefetching strategy.
- Overhead: Ratio of videos being prefetched and not requested by the user divided by the number of requests that were not served from the user's local cache. Formally, it is equal to:

$$Overhead = \frac{(1 - CPR) * N_{\text{prefetched videos}}}{(1 - HR) * N_{\text{requested videos}}} \tag{4}$$

- False Negative Ratio (FNR): Ratio of requests that the prediction policy failed to detect, although clients have already been notified about these contents: The QNotif holds an index pointing to the requested content, but the content was not prefetched since it was not considered to be relevant for the user. The reason for this failure might be that the content was not considered popular enough $(p_v < p_{th})$, and/or the freshness of the content was not good enough to position it among the N_{prefetch} best ranked content items in QNotif, i.e. among the ones selected as prefetching candidates.

[2] http://www.ict-ecousin.eu/public-deliverables-dissemination/public-deliverables/ecousin-deliverable-d3.2-v1.0-public.pdf/view

Thus the sum of FNR and CPR gives insights into the optimal CPR we may achieve.

The goal of the Prefsim implementation is to get insights into the prediction accuracy we may achieve.

A. Simulation setup

In Prefsim, the simulation setup is defined in an XML-based configuration file (input.xml). We evaluated our system using the FB dataset that was introduced in Section I. In order to ensure fast simulation processing, we only considered the users who requested at least 100 videos over the whole data collection period (in average 1 video per day). Table II summarizes the traffic trace used for the simulation. The sparsity level $(1 - \frac{\text{number of requests}}{\text{number of user} * \text{number of videos}})$ is extremely high and, thus, sticks to the real world traffic properties.

The interest graph is updated in a daily routine. Each user has at most 20 similar neighbors. The cache size used for users is limited to 50 videos and LRU is used as a cache replacement policy. We maintain a history size of the 10 past requests to model the user activity. r is equal to 0.334. For a given user, we do not prefetch videos unless he requests at least 20 videos. This is to get a first insight on users' preferences and to better assign neighbors for each user in the system. The sleep state period (Δ) is set to 1 hour. Regarding the configuration of the queues QNotif and QCandidates, Γ is set to 3 days, and only the videos that belong either to the classes C_0 or C_1 of QNotif are candidates for prefetching.

Number of FB sessions	Number of unique users	Number of unique videos	Sparsity level
4,152,885	23,548	962,406	0.9998

TABLE II. TRAFFIC TRACE USED FOR THE SIMULATION

B. Performance analysis

In this part, we evaluate CPSys by conducting two experiments. In the first experiment, we evaluate several prefetching policies and show how MPMR outperforms other policies, while in the second exeperiment we compare CPSys to a naive prefetching strategy where we catch the most viewed videos of the day and push them to all clients.

a) MPMR Evauation: In the first experiment, we evaluate the average CPR and the overhead of several policies adopted by QNotif: FIFO, LRU, and MPMR-p_{th}, i.e. the MPMR policy with different values for the threshold popularity score p_{th} ($p_{th} \in [1..5]$). In the same experiment, we also compare the case where $N_{prefetch}$ is equal to either S_{max} or to 6. Figure 7 shows that regardless the values of p_{th}, MPMR outperforms the LRU and FIFO policies. We observe a significant increase in the CPR when $p_{th} > 1$, reaching up to 18%. The CPR improves when the value of p_{th} increases, as this makes the prediction policy more conservative: the prefetching process is triggered only when the content was viewed at least p_{th} times by the neighbors.

Figure 8 shows that the overhead decreases significantly when adopting MPMR, especially when p_{th} is high. We observe that if prefetching is not well tuned, then the traffic increases significantly and may even double, which obviously cannot be accepted by Telcos. However, a fine configuration of parameters significantly decreases the Overhead. We show that it drops below 5% when $(p_{th} > 2)$. Besides, adapting the number of prefetched videos with respect to the past user activity $(N_{prefetch} = S_{max})$ also reduces the overhead, hence leads to a better network experience.

The lesson we learn from this result is in line with Finding F1: it is safer to avoid prefetching very unpopular contents and wise to control the overhead caused by content prefetching on the network load.

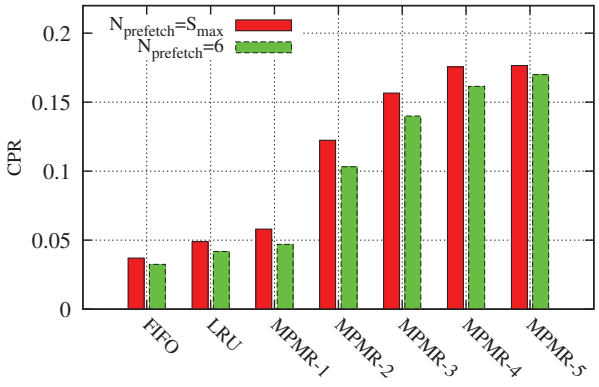


Fig. 7. CPR with different content seleciton policies

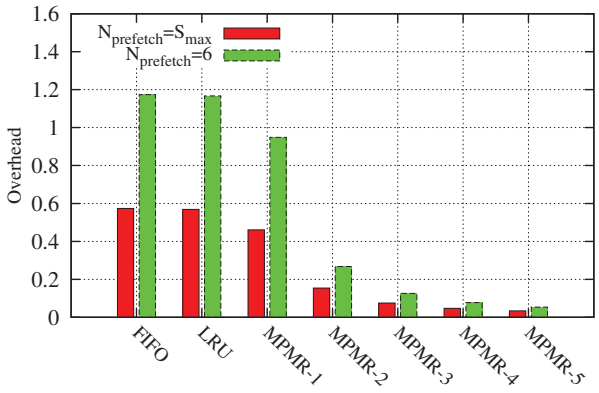


Fig. 8. Overhead with different content selection policies

Figure 9 shows that the FNR increases as far as the prediction policy becomes more conservative, which in return increases the CPR and decreases the overhead. Relying on users' neighbors, FNR reaches 12% and CPR reaches 18% when p_{th} is equal to 5, this suggests that while setting p_{th} to 5, an optimal content selection policy would rise CPR to 30%. However, decreasing the FNR will systematically increase the overhead. This trade-off should be carefully handled in operational networks.

The lesson we learn from the 3 past simulation results is that whatever the policy used to select the prefetching candidates, prediction is still hard in general. While the MPMR suggests pushing the most popular and fresh content items that have been seen by the most similar neighbors, we observe that the large majority of viewed videos by individual users

are considered as personalized content items. Prefetching these personalized content items is risky and leads to an acute trade-off between the overhead and prediction accuracy. In CPsys, the more we learn about users' preferences, the more accurate the prediction model is. Unfortunately, the dataset we used does not cover all users' preferences since it was collected from mobile networks. We could have successfully pushed a content the user is interested in. However, this user was connected to a fixed network through a WiFi connection the time he requested the content. In this case, we do not capture this request in our mobile traffic traces. This suggests that the performance assessment we carry out in this section represents the lower bound of the real performance we may achieve.

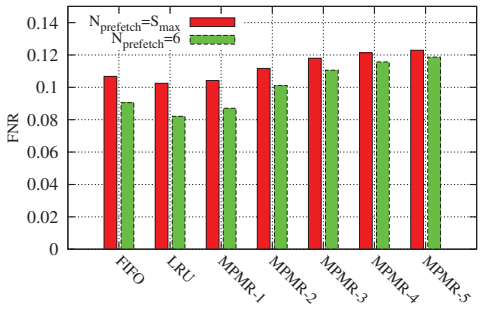


Fig. 9. FNR with different content selection policies

b) Prefetching local VS global popular videos: In the second experiment, we limit the simulation to 4 days and compare our prefetching system with the authors' proposition in [10] which consists of pushing bundles of popular contents on mobile devices. Mapping this to Prefsim, if one user watches a video, then every user should update his list of prefetching candidates with the most recently watched video. Figures 10 and 11 show that limiting the neighborhood to the 20 most similar users and setting p_{th} to 3 improves the CPR up to 3 times, while it decreases the overhead by 5 times. This means that pushing only popular contents to everyone is not necessarily the best option with regard to the prediction efficiency. Yet, it is better to personalize the list of prefetching candidates according to the preferences of the user's most similar neighbors.

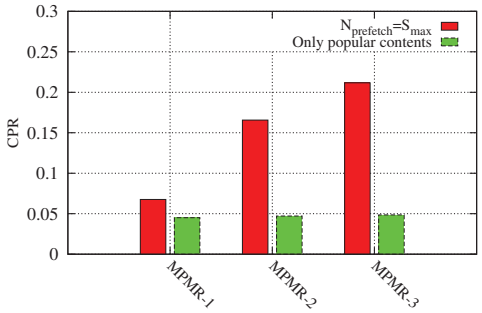


Fig. 10. CPR

VI. Prototype Implementation

We have implemented a prototype of CPSys using a client-server model. The central predictor runs as a third-party server

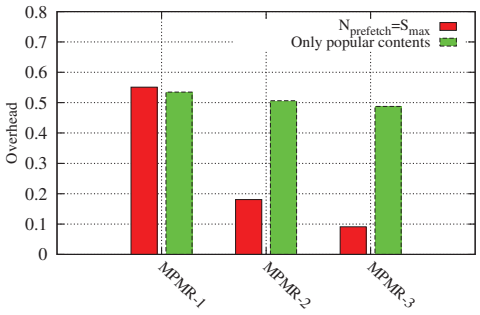


Fig. 11. Overhead

holding profiles of all users running the prefetcher agent.

At the server side, we used an Apache Tomcat server [3] as well as the Jersey framework [4] to implement the restful web services. We used the Mahout framework [5] to build and update the interest graph and compute the Jaccard affinity scores. As a content selection policy, we implemented MPMR.

At the client side, users should install the CPClient which is an Android-based application. The CPClient represents the prefetching agent with a frontend interface (c.f. Figure 12(a)). The CPClient is linked to the user's Facebook account, i.e. the user is asked to log in to his Facebook account to initiate running the CPClient. The reason for this is that, in CPSys, we use the Facebook_ID as a unique user identifier. The same ID is used at the server to update all data structures, including databases, QNotifs, QViewed.

We use an intent-filter mechanism [6] to follow users' activities and report the list of videos watched to the CPserver. When prefetching is executed, CPClient pulls the video IDs from QNotif. Subsequently, the agent asynchronously prefetches these videos. When prefetching is complete, thumbnails of the videos are displayed (cf. Figure 12(b)). In this example, QNotif holds the IDs of YouTube videos which are considered for prefetching.

At last, in our prototype implementation, we enriched the prefetching-app with a notification mechanism (c.f. Figure 12(c)). When the prefetching process is over, a user – running CPClient receives a notification to motivate him or her to watch the recently prefetched videos. We believe that this incentive strategy improves both CPR and HR, since clients are more likely to consult and watch the prefetched videos.

A video demonstration of this work is available at http://tinyurl.com/pq2v28s.

VII. Conclusion

In this paper, we designed, evaluated and implemented CPSys, a prefetching system we have designed based on traffic patterns and clients' behavior that we observed in a real operational mobile network. We addressed a series of key design issues. Subsequently, CPSys relies on recommendation

[3] http://tomcat.apache.org/
[4] https://jersey.java.net/
[5] https://mahout.apache.org/
[6] http://developer.android.com/guide/components/intents-filters.html

(a) CPClient interface

(b) Prefetched videos

(c) Notification

Fig. 12. Snapshots from CPClient

techniques to build the implicit or social graph, then we use the MPMR policy to select the video prefetching candidates. At the end, we evaluated CPSys through trace-driven simulations. We show that the highest lower-bound performance of CPSys regarding CPR ranges from 18% to 22% while we show that the traffic overhead decreases significantly. We observed that prefetching performance is strictly related to content characteristics. When content items become further personalized, prediction becomes harder and potentially reduces the prediction accuracy.

There are many possible venues towards enhancing CPSys. One primary direction would be to focus on the personalized content items which are the key driver for the long tail distribution, hence supplying CPSys with additional information to further personalize the list of prefetching candidates. In parallel, we plan to enhance our system implementation and make it faster and more scalable. The goal is to study how CPSys performs at large scale and study how notifications can improve the system performance.

REFERENCES

[1] M. Z. Shafiq, L. Ji, A. X. Liu, and J. Wang, "Characterizing and modeling internet traffic dynamics of cellular devices," in *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*. ACM, 2011, pp. 305–316.

[2] P. Gill, M. Arlitt, Z. Li, and A. Mahanti, "Youtube traffic characterization: a view from the edge," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. ACM, 2007, pp. 15–28.

[3] T. Bucher, "Want to be on the top? algorithmic power and the threat of invisibility on facebook," *New Media & Society*, vol. 14, no. 7, pp. 1164–1180, 2012.

[4] Y. Li, Y. Zhang, and R. Yuan, "Measurement and analysis of a large scale commercial mobile internet tv system," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM, 2011, pp. 209–224.

[5] Z. Li, J. Lin, M.-I. Akodjenou, G. Xie, M. A. Kaafar, Y. Jin, and G. Peng, "Watching videos from everywhere: a study of the pptv mobile vod system," in *Proceedings of the 2012 ACM conference on Internet measurement conference*. ACM, 2012, pp. 185–198.

[6] X. Cheng and J. Liu, "Nettube: Exploring social networks for peer-to-peer short video sharing," in *INFOCOM 2009, IEEE*. IEEE, 2009, pp. 1152–1160.

[7] Z. Li, H. Shen, H. Wang, G. Liu, and J. Li, "Socialtube: P2p-assisted video sharing in online social networks," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 2886–2890.

[8] N. Gautam, H. Petander, and J. Noel, "A comparison of the cost and energy efficiency of prefetching and streaming of mobile video," in *Proceedings of the 5th Workshop on Mobile Video*. ACM, 2013, pp. 7–12.

[9] P. Mohan, S. Nath, and O. Riva, "Prefetching mobile ads: Can advertising systems afford it?" in *Proceedings of the 8th ACM European Conference on Computer Systems*. ACM, 2013, pp. 267–280.

[10] A. Finamore, M. Mellia, Z. Gilani, K. Papagiannaki, V. Erramilli, and Y. Grunenberger, "Is there a case for mobile phone content pre-staging?" in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*. ACM, 2013, pp. 321–326.

[11] F. Figueiredo, J. M. Almeida, M. A. Gonçalves, and F. Benevenuto, "On the dynamics of social media popularity: a youtube case study," *ACM Transactions on Internet Technology (TOIT)*, vol. 14, no. 4, p. 24, 2014.

[12] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I tube, you tube, everybody tubes: analyzing the world's largest user generated content video system," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. ACM, 2007, pp. 1–14.

[13] R. I. Chiang, G. B. Rowe, and K. W. Sowerby, "A quantitative analysis of spectral occupancy measurements for cognitive radio," in *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th*. IEEE, 2007, pp. 3016–3020.

[14] S. Shafer and D. Rogers, "Similarity and distance measures for cellular manufacturing. part i. a survey," *THE INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH*, vol. 31, no. 5, pp. 1133–1142, 1993.

[15] A. Gouta, D. Hong, A.-M. Kermarrec, and Y. Lelouedec, "Http adaptive streaming in mobile networks: characteristics and caching opportunities," in *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on*. IEEE, 2013, pp. 90–100.

[16] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 233–240.