O^2SM: Enabling Efficient Offline Access to Online Social Media and Social Networks

Ye Zhao[1], Ngoc Do[1], Shu-Ting Wang[2], Cheng-Hsin Hsu[2], and Nalini Venkatasubramanian[1]

[1] Department of Information and Computer Science, University of California, Irvine, USA
[2] Department of Computer Science, National Tsing Hua University, Hsin-Chu, Taiwan

Abstract. In this paper, we consider the problem of efficient social media access on mobile devices, and propose an Offline Online Social Media (O^{2}SM) Middleware to: (i) rank the social media streams based the probability that a given user views a given content item, and (ii) invest the limited resources (network, energy, and storage) on prefetching only those social media streams that are most likely to be watched when mobile devices have good Internet connectivity. The ranking scheme leverages social network information to drive a logistic regression based technique that is subsequently exploited to design an utility based content prefetching mechanism. We implemented O^{2}SM and a corresponding app, oFacebook, on Android platforms. We evaluated O^{2}SM via trace data gathered from a user study with real world users executing oFacebook. Our experimental results indicate that O^{2}SM exhibits superior viewing performance and energy efficiency for mobile social media apps; its lightweight nature makes it easily deployable on mobile platforms.

1 Introduction

The last few years have witnessed the wild success of social networks and social media sites such as Facebook, Twitter, LinkedIn, Google+, and Instagram – communications using these networks have indeed become a part and parcel of our lives today. With the growing popularity of mobile communications, users connect to these social networks using mobile devices (e.g., smartphones, tablets) and expect to derive the same experience no matter how and where they connect. A market study reports that 37% of users accessing rich content, e.g., video, audio, and images, on PCs have switched to mobile devices; moreover, 55% of smartphone users use Facebook on their smartphones [4]. Another survey, done in Canada, shows that 61% of users access Facebook through mobile devices [1]. These market reports indicate that social media streams, carrying rich content, have become key aspects of how people view information and how society interacts today. However, several challenges arise in enabling rich social media on mobile devices including: (i) sporadic network availability causing intermittent access (ii) bandwidth limitations in shared wireless media, and (iii) high access cost from volume driven dataplans – all of the above result in degraded user experience and limit the exchanges of social media information on mobile devices.

A simple approach to cope with intermittent Internet access is to share or download social media streams when possible, as if they were traditional media streams from content providers, such as Netflix and Hulu. However, there are key differences between social media and traditional media streams that makes best effort downloads have inferior performance in our case:

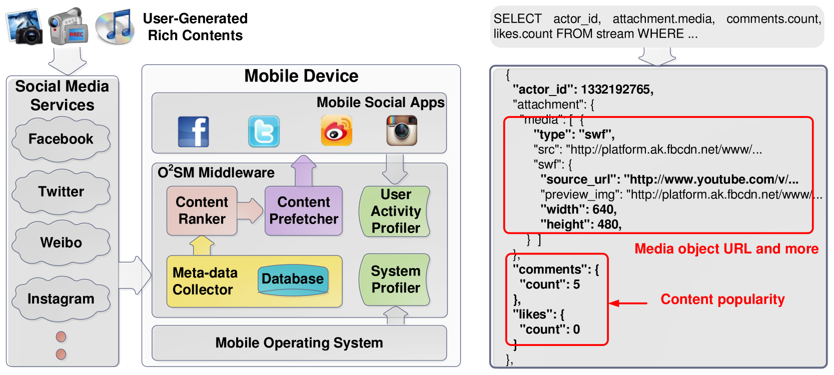
- **Personalized preferences:** The demands of traditional media streams are dictated by global popularity (e.g., movies, television), whereas access to rich social media streams are driven by user preferences and content characteristics. Therefore, traditional one-size-fits-all, popularity based content ranking mechanisms do not work for social media streams.
- **Dynamic, variable size contents:** Compared to traditional media, social media sharing is exemplified by more dynamic uploads (by users); the average size of content that is shared during an update is also smaller than that of media content providers. For example, it is reported that majority of YouTube videos are shorter than 10 mins, while traditional video servers offer up to 2-hour videos [30].
- **Sporadic viewing situations:** Dynamic uploads by social network users and asynchronous access (by their friends) result in sporadic access patterns, e.g., users checking Facebook updates at a cafeteria line. Unlike media downloads where the item of interest can be pre-specified by the users, users may wish to view newly uploaded content in new situations, including those where mobile Internet access is unavailable, intermittent, or expensive (e.g., on a bus/train).

Today, the social media mobile applications (apps), such as mobile Facebook [2], are simple client side implementations that work as dedicated social media browsers. In the absence of connectivity at the time users wish to use the apps, users do not have access to up-to-date social media content. In this paper, we consider the problem of efficient social media access on mobile devices and develop solutions to make the mobile social media experience seamless, personalized, and effective. We achieve this by: (i) *ranking* the social media streams based the probability that a given user views a given content item, and (ii) investing the limited resources (network, energy, and storage) on *prefetching* only those social media streams that are most likely to be watched when mobile devices have good Internet connectivity.

A comprehensive solution for these two tasks must account for multiple factors including network conditions, content characteristics, device status, and users' social networks. These factors are typically outside the purview of content providers (e.g., current network conditions for a particular user) or the app at the user side (e.g., characteristics of a specific content item). We develop a mobile middleware system that bridges the device OS and network environment to various social networking apps, such as Facebook, Google+, LinkedIn, and Twitter to determine what content to prefetch and when. Such a middleware approach allows us to: (i) reuse key techniques and implementations across multiple devices/users (ii) make better prefetch decisions based on global information, and (iii) control the overhead due to data transfers and environmental sensing. Note that the proposed schemes are designed to work in tandem with user driven navigation of social media content (e.g., through mobile Facebook) and not replace them. In fact, we envision that users can provide feedback on relevance of the content through existing well known frontends (such as mobile Facebook).

The key contributions of this paper are as follows:

- We design a comprehensive *Offline Online Social Media (O^2SM)* middleware to enable accessing social media on mobile devices with intermittent Internet access.



(a) The proposed system architecture

```
SELECT actor_id, attachment.media, comments.count,
likes.count FROM stream WHERE ...
```

```
{
  "actor_id": 1332192765,
  "attachment": {
    "media": [ {
      "type": "swf",
      "src": "http://platform.ak.fbcdn.net/www/...
      "swf": {
        "source_url": "http://www.youtube.com/v/...
        "preview_img": "http://platform.ak.fbcdn.net/www/...
        "width": 640,
        "height": 480,
    } ]
  },
  "comments": {
    "count": 5
  },
  "likes": {
    "count": 0
  }
},
```

(b) Example of social media stream meta-data

Fig. 1. The proposed system architecture and an example of social media stream metadata.

– We develop, implement, and evaluate novel algorithms for content ranking and prefetch planning to optimize the user experience of mobile social apps. The proposed techniques are theoretically grounded using machine learning and optimization based approaches.
– We implement a mobile middleware system and a concrete app, oFacebook (offline Facebook), to automatically prefetch user-generated contents in social media streams to provide mobile users uninterrupted access to social networks.
– We conduct a user study using oFacebook to collect trace data from 10 users, to determine how users access media information on Facebook via mobile devices. We perform extensive trace-driven simulations to evaluate the proposed middleware.

To the best of our knowledge, O^{2}SM is the first system that enables a systematic, automated approach for viewing relevant and up-to-date social media in an offline manner.

2 System Architecture

Fig. 1(a) presents the architecture of our proposed middleware, O^{2}SM that bridges the mobile OS and social network apps running on a user's mobile device. O^{2}SM systematically detects user-generated rich content from the mobile user's social media streams (e.g., fresh photos or videos posted by the user's friends), and intelligently prefetches contents in order to cope with intermittent Internet access. The choice of what to prefetch and when is crucial since the mobile device is limited in resources, such as battery level and storage space - in the social media context, it is imperative that the mobile user's demands and preferences are taken into consideration while selectively downloading content. Furthermore, energy efficient download is also a must in order to achieve good user experience, e.g., overly-aggressive prefetching decisions may drain the device battery , which prevents the user from using his or her phone for necessary daily activities.

The O^{2}SM middleware consists of five components: (1) *system profiler*, (2) *user activity profiler*, (3) *meta-data collector*, (4) *content ranker*, and (5) *content prefetcher*. The system profiler monitors network and battery conditions on the device via various performance metrics, including signal strength, network throughput, and battery level. It also monitors the size of total downloads to avoid filling up the storage space. Based on the current network conditions and past history of network profiles, the system profiler forecasts the network connectivity and throughput in the near future. The user activity profiler monitors the user's activities on the device. In particular, it monitors the user's access patterns to social media streams, e.g., when and how long the user navigates the social media streams, and forecasts the user's future accesses. This information can be collected from the history log of the social media apps built atop the middleware or upon explicit user input. For example, if the user plans to go out for lunch in an hour, the user can request the middleware to aggressively prefetch social media contents before, so that he/she has something to watch while waiting in the restaurant.

The meta-data collector periodically pulls the meta-data from each of the social media streams to get updates of new rich contents since the last pull. This is done when the network is available and the remaining battery level is above an operational threshold, which is configurable by the user. The meta-data is usually small in size. Fig. 1(b) shows an example meta-data request and its JSON responses for a media content using Facebook Query Language (FQL) [3]. Our experiments show that the average meta-data size is around 700 bytes per content, and is negligible when compared with the typical data size of 140 KB of a photo or that of 3 MB of a 1.5-minutes video. The collected meta-data is saved onboard the meta-data database for later processing.

To support intelligent content prefetching, the content ranker invoked by the content prefetcher queries the database on fresh content items that have not been downloaded by the system as yet, and predicts the probability the user would view them. The ranking algorithm is based on social relations between entities in the social networks, user interests and content popularity. The ranked contents are fed to the content prefetcher to make decisions on which contents to prefetch and when to prefetch. The crux of this component is a cost-benefit driven prefetch scheduling algorithm which aims to maximize the overall prefetching gain (benefit-cost) by intelligently selecting contents to download and scheduling their download time. This is achieved by carefully taking into account the variations in the sizes of the contents (from content meta-data), their expected probability of being watched (from the content ranker), predicted network conditions such as the network connectivity and bandwidth (from the system profiler), as well as predicted user activity such as when the user will navigate his/her social media streams (from the user activity profiler). Once the contents are scheduled for downloads, they are inserted into a download queue. The downloader downloads the contents in the queue based on their scheduled times.

3 Social Media Content Ranking Component

Given resource constraints at mobile devices, a method to efficiently rank social media contents based on their importance is critical. Prefetching all media contents that accompany heavy multimedia objects, e.g., images or videos, could not only be harmful to user, when he or she is in need of phone battery to survive in daily work, but also waste resources if the downloaded contents are not later viewed by user. This section is dedicated to present the background and our approach to develop a ranking system that works efficiently for mobile devices and scalably against the arrival of new contents.

Background: Traditional ranking systems can be classified into three categories: (1) content based approaches [5,9,12], (2) collaborative filtering approaches [7,20,27], and (3) social network based approaches [19,21,25]. Content based approaches [5,9,12] rank media contents using the correlation between the attributes of media content, such as descriptions, keywords, tags, images, and video features (e.g., colors), and user preferences. The basic process here matches up the attributes of the user profile (with preferences,interests), with the attributes of contents. The performance of this category is limited by dictionary-bound relations between the keywords obtained from users and the descriptions of media contents. More comprehensive feature extraction (e.g., image and video analysis) is expensive and hence not supported by most social media.

Collaborative filtering approaches [7, 20, 27] recommend media contents by first calculating the similarity between all users in the system based on their previous ratings of media contents. Ratings are, for example, represented using numeric scores from 1 to 5 and similarity is estimated using heuristic measures, such as the well known cosine function. The system, then, projects a ranking a user is likely to give a piece of recommended content by aggregating the ratings of the user's k nearest neighbors on the content. Collaborative filtering approaches are useful for highly sparse data where the matrix (ratings) is partially missing. Issues for these approaches include *cold start*, i.e., all ratings are missing, and high computation complexity, i.e., a large number of users and contents involves an inference process, that is critical due to limited resources on mobile devices.

The last category ranks social media contents via social networks. Social networks are characterized by heterogeneous entities, e.g., contents, users, and social relations. Recent studies [19,21,25] exploit relations among entities to recommend contents. For example, a user is likely to view contents generated by friends, whom the user interacts with frequently over social networks. The approaches in this category fit our system's goals best because they could utilize the underlying social relations captured in our system. However, while the intuitions are appealing, a direct application of the existing techniques to our system is not easy. For example, the work in [19] is based on a social graph to evaluate social relations among entities, and works only for a fixed set of entities. It requires rebuilding the social graph and recalculation of weights when new contents arrive, which happens frequently in our system. Extensions to the collaborative filtering concept [21, 25] by employing relations to estimate similarity have been studied, they suffers from high computation complexity that are unsuitable to mobile deployment.

Our Approach: We inherit the spirit of the social network based recommendation systems [19,21,25] to rank social media contents based on social relations and interactions, but targets to work efficiently on mobile devices and be scalable to the arrivals of new contents. Besides, we take user-poster interests and content popularity into account to improve the ranking accuracy. In some sense, user-poster interests and content popularity can be considered as indirect content based features because they capture user preferences and content impacts on viewers. The main design principles for the ranking component in our mobile system are *light weight* and *fast provisions of ratings on newly incoming contents*.

Our approach first identifies and constructs social based features that can infer user-content interactions. We then employ a supervised learning approach to predict the probability of a social media content viewed by a user. In this paper, we develop our

Interaction Type	Explanations
Post(u, f)	User u writes a post on poster f's Wall page, tags f in a photo, a video or an album of images.
Post(f, u)	Vice versa.
Comment(u, f)	User u comments on a post from f, or on a tag of a photo, a video or an album from f.
Comment(f, u)	Vice versa.
Like(u, f)	User u likes a post uploaded by f, or an image, a video or an album tagged by f, or u likes a comment from f.
Like(f, u)	Vice versa.
Message(u, f)	User u sends a private message to f.
Message(f, u)	Vice versa.

Table 1. Representation of user interactions.

ranking component to support content ranking on Facebook, but the approach is general enough to be readily applied to other social networks, e.g., Twitter as presented in [13].

We construct the following features to capture the underlying social relations. Let u be a user with a friend f. Table 1 presents key interaction types between u and f.

1. **Post interactions**: This feature captures the interactions between the user and a friend of the user via posts on their social media sites, e.g., Facebook Wall page. It is clear that u is likely more interested in f's posts if interaction between u and f is high. We use the total number of interactions between u and f to quantify this feature. Post interactions include post(u, f), post(f, u), comment(u, f), comment(f, u), like(u, f), and like(f, u) as shown in Table 1. In Facebook, a user can post message on his or her friend's Wall page or tag the friend in a photo. A Twitter user can retweet an interesting message or @reply a tweet from some person the user follows. All of these are considered post interactions. Some social networks, for example Facebook, allow users to subscribe or like a page (e.g., a soccer club like Manchester United or a university like Harvard University). Any updates from the page will be sent to the subscribers similar to those from a friend. For simplicity, we consider a page as a friend of u.
2. **Private message exchange**: Different from post interactions, which may be available in public, private messages are only available to the recipients. We use the total number of messages sent and received between user u and friend f of u to quantify the feature (i.e., message(u, f) and message(f, u) in Table 1). The higher number of exchanged messages, the higher the interaction level between u and f and higher the probability that u views f's posts. In Facebook, users can send and receive private email-like messages l to and from their friends (and even strangers). Twitter and Weibo users can communicate with each other through direct messages with a limited number of characters.

In addition to the above features, we extract and represent two others features - the level of interest of a user u has to a specific friend f, and the popularity of a post.

1. **User interest w.r.t to a friend**: We measure the interest of u w.r.t to a friend f of u by the number of view clicks to contents posted by f. The higher number of view clicks indicates a higher interest level of u to f.

2. **Post popularity**: A post from friend f,with a high number of comments and likes, is likely interesting because it receives high attention from many viewers. It is therefore likely to be interesting to u even the viewers are not u's friends. We thus employ the number of comments and likes in a post to predict the viewing probability.

In order to predict the probability a content is viewed by a user, we employ a well-known supervised learning algorithm - logistic regression classifier. Logistic regression is an additive model used to predict the probability of a discrete event given a set of explanatory variables. It weights the impact of all constructed features with estimated coefficients. In the literature, the logistic regression has been used for prediction, such as friendship strength prediction [16]. We are the first applying the logistic regression to predict the probability of user-content interactions based on the social networks.

Now we describe how to apply the logistic regression to our content ranking component. The logistic regression in the training process learns a model of the form:

$$p(y_i|x_i) = \frac{1}{1 + e^{-(c_0 + \sum_1^n c_j x_i^j)}}, \tag{1}$$

where x_i is a vector of the features $(x_i^1, ..., x_i^n)$ described above for content i, and y_i is 1 if the user clicks to view content i. Otherwise, y_i is 0. The model is represented by a vector of coefficients c, in which c_0 is not multiplied with any feature and added to the vector as noise. In the training process, the model is learned by maximizing the log-likelihood of the logistic regression model. A well-known approach for the maximization problem is a trust region Newton method [18]. The resulting model is then used to evaluate the viewing probability for a newly arriving media content by simply applying Eq. (1) to the features of the new content. In the evaluation section, we will show that the training process that calculates the model parameters for our ranking component is indeed light weight and fast. This technique is attractive in our setting since the processing of newly arriving contents essentially reduces to that of applying Eq. (1). Furthermore, the simplicity of the scheme lends itself to be easily deployed on mobile devices.

4 Ranking-driven Social Media Prefetching

Prefetching social media contents has received attention at CDNs. [29] considered the problem of efficient geo-replicating contents across multiple data-centers spread around the world. However, little research has been done for prefetching social media contents to mobile clients. Mobile prefetching is an increasingly relevant topic of research. Techniques have been developed for determining when to prefetch based on network conditions such as WiFi and cellular signal strength [10, 26, 28]. Minimizing energy consumption during prefetching is critical for mobiles - studies [6, 11] indicate that WiFi is typically more efficient than cellular networks, and techniques to aggregate multiple data transfers to save energy have been developed. [8, 17] exploit social networks to assist prefetching video prefixes. In general, prefetching has been explored in client-server and peer-to-peer settings using factors such as energy, transfer volume and user preferences.

Much of the earlier work focuses on what to prefetch and does not explicitly account for whether the prefetched content is consumed or not. A more comprehensive

```
Algorithm 1: Prefetch Scheduling
    Step 1-Read Unviewed Ranked Contents:
    read the list of unviewed ranked contents output from the Content Ranking component.
    Step 2-$K$ Slot-ahead Forecast:
    forecast the network conditions and user context for the next $K$ slots.
    Step 3-Offline Online Social Media Prefetch Scheduling (O$^2$SMPS):
    formulate an O$^2$SMPS problem using a cost/benefit analysis to decide which contents to
    download in each of the next $K$ slots.
    Step 4-Contents Download:
    sequentially download contents that are scheduled for the current slot.
```

prefetching scheme must incorporate *which* items to prefetch for maximum benefit. [31] observes that users tend to launch a fairly fixed sequence of the apps, and user locations have a strong correlation with app usage so a decision engine was proposed to determine which apps to prefetch. In contrast to the above approaches, we propose a comprehensive scheme that exploits the previously described ranking mechanism to determine *what* to prefetch and develops a efficient scheduling technique for *when* to prefetch.

The O^{2}SM prefetcher divides time into *prefetching period* or slot (e.g., 15 minutes), say $T_{prefetch}$, and runs the prefetch scheduling algorithm at the start of every slot. The algorithm selects a number of contents for download in each prefetching period optimizing for large time-scale prefetching. Algorithm 1 shows the flow of the scheduling algorithm. The goal of the algorithm is to provide the best viewing experience when the user navigates his/her social media streams, while keeping the prefetching cost low. The O^{2}SM prefetcher strives to balance the potential benefit of prefetching against its cost: Since the prefetching benefit is different for different contents and the prefetching cost of the same content varies over time when network condition changes, the prefetch scheduling algorithm determines what to download and when to download by formulating a *Offline Online Social Media Prefetch Scheduling (O^{2}SMPS)* problem. In the following sections we describe the cost/benefit modeling and the O^{2}SMPS problem in details, and discuss the forecasting techniques used in our system.

4.1 Cost/Benefit Modeling

Let $I = \{i_1, i_2, \ldots, i_{|I|}\}$ be the list of ranked contents and q_i is the likelihood of content i will be viewed by the user. We evaluate the cost/benefit of content prefetching in the next K slots to determine the best prefetching time by forecasting the network conditions and user activity in the future. Let $t = 1, 2, \cdots, K$ be the slotted time for the next K prefetching slots. The system monitors the use of the 3G and WiFi network interfaces for Internet access. Let $p_{wifi}(t)$ be the probability that the device uses the WiFi interface for data transfer at slot t, and $bw_{wifi}(t)$ is the corresponding download bandwidth. Similarly, $p_{cell}(t)$ and $bw_{cell}(t)$ are the probability and bandwidth for the 3G interface at slot t. The system also predicts how likely the user will actively navigate his/her social media streams. Let $p_{nav}(t)$ denote the predicted likelihood that the user may become active at slot t. We discuss the network and user activity profiling and prediction techniques in the Sec. 4.2.

Inspired by [14], we estimate the cost and benefit of prefetching each content along three dimensions: *viewing performance*, *energy use*, and *data plan use*. We consider two costs involved in prefetching: the *energy cost* and the *cellular data plan cost*. The cost for prefetching a content may vary over time when the network conditions change. Formally, we define the estimated cost of prefetching a content i at slot t_{pre} as:

$$C(i, t_{pre}) = w_e \times C_{Energy}(i, t_{pre}) + w_d \times C_{cell}(i, t_{pre}),$$

where w_e and w_d are weighting coefficients for energy and cellular data consumption respectively. The coefficients are configurable to the user. For instance, if the user uses an unlimited cellular data plan so that the cellular data usage is not a concern, then w_d can be set to zero. On the other hand, if the user is discreet about energy use, he may prefer a large value for w_e.

The benefit of prefetching is threefold: the *viewing performance benefit*, the *energy benefit* and the *cellular data plan benefit*. The viewing performance benefit comes from the improved user experience when the user views a prefetched content. The energy and the data plan benefits are the saves in the energy use and data plan use that would be otherwise consumed when the user views the content. Clearly, the benefit depends on: (i) probability of the content to be viewed and (ii) network condition at the time when the user navigates the streams. Formally, we define the estimated benefit to prefetch a content i in case the user navigates his/her social media streams in slot t_{nav} as:

$$B(i, t_{nav}) = B_{view}(i, t_{nav}) + w_e \times B_{energy}(i, t_{nav}) + w_d \times B_{cell}(i, t_{nav}).$$

We next define $B_{energy}(\cdot)$. We adopt the energy models developed by PowerTutor [32]. For downloading a content under WiFi at slot t, the energy cost is modeled as $e_{wifi}(i, t) = c_{wifi} \times \frac{s_i}{bw_{wifi}(t)}$, where c_{wifi} is a power coefficient for WiFi interface. For downloading under 3G, the energy cost is $e_{cell}(i, t) = c_{cell} \times \frac{s_i}{bw_{cell}(t)} + e_{tail}$, where c_{cell} is a power coefficient for 3G and e_{tail} is an estimated 3G tail energy cost. Since contents are prefetched in batches, to estimate the tail energy cost we let $e_{tail} = c_{tail} \times \frac{T_{tail}}{l_{avg}}$ where c_{tail} is the power coefficient for 3G tail energy, T_{tail} is the typical 3G tail time (usually > 10 seconds), and l_{avg} is the history average of the number of contents downloaded in a batch by the prefetcher. To predict the 3G tail energy consumption for on-demand fetches from the user, we let $e_{tail} = c_{tail} \times \min(T_{inactive}, T_{tail})$ where $T_{inactive}$ is the history average of the idle period between two consecutive content requests from the user. Then, the expected energy to download a content i at slot t is:

$$E(i, t) = \frac{p_{wifi}(t)}{p_{wifi}(t) + p_{cell}(t)} \times e_{wifi}(i, t) + \frac{p_{cell}(t)}{p_{wifi}(t) + p_{cell}(t)} \times e_{cell}(i, t).$$

Since the prefetching energy cost $C_{energy}(i, t_{pre}) = E(i, t_{pre})$, the energy benefit becomes $B_{energy}(i, t_{nav}) = q_i \times E(i, t_{nav})$, which takes the probability of the content to be viewed into consideration.

We define $B_{cell}(\cdot)$ in a similar way. The expected cellular data plan use for downloading a content i at t is:

$$D(i, t) = \frac{p_{cell}(t)}{p_{wifi}(t) + p_{cell}(t)} \times s_i.$$

Therefore, the data plan cost $C_{cell}(i, t_{pre}) = D(i, t_{pre})$ and data plan benefit $B_{cell}(i, t_{nav}) = q_i \times D(i, t_{nav})$.

The viewing performance benefit is considered as the granted feasibility for offline access if the user desires to view a content when he/she doesn't have network access. On the other hand, if the user checks the content when he/she has network access, the benefit is the hidden latency of the on-demand content downloading/buffering. Assume the download bandwidth at the time when the user requests the content is bw, the hidden latency can be formulated as $d(i) = \frac{\min(s_i, s_{max})}{bw}$, where s_{max} is the maximum playback buffer size for videos, or $s_{max} = s_i$ for non-video content. Then, taking into account the viewing probability of the content as well as the predicted network conditions, for any time slot t_{nav} that the user may actively navigate social media contents, the expected viewing performance benefit for prefetching content i is:

$$\begin{aligned} B_{view}(i, t_{nav}) = p_{wifi}(t_{nav}) \times q_i \times d_{wifi}(i, t_{nav}) + p_{cell}(t_{nav}) \times q_i \times d_{cell}(i, t_{nav}) \\ + w_{off} \times (1 - p_{wifi}(t_{nav}) - p_{cell}(t_{nav})) \times q_i, \end{aligned}$$

where $d_{wifi}(i, t_{nav})$ and $d_{cell}(i, t_{nav})$ are the predicted hidden latency under WiFi and 3G respectively, and w_{off} is a relative benefit weight of viewing offline to viewing online, that can be customized by the user.

4.2 O^{2}SMPS Problem

The O^{2}SMPS problem maximizes the prefetching gain as the benefit minus cost, by allocating contents for downloads in each of the next K slots. We define a prefetch scheduling matrix, $Z = \{z_{i,k}\}$, where $z_{i,k} \in \{0, 1\}$, such that $z_{i,k} = 1$ if download content i at slot k and $z_{i,k} = 0$ otherwise. Furthermore, let $y_i(k)$ be another 0-1 variable that indicates whether a content i *has been* downloaded before slot k. It is easy to see that $y_i(1) = 0$ and $y_i(k) = \sum_{l=1}^{k-1} z_{i,l}$ under the constraint that $\sum_{k=1}^{K} z_{i,k} \leq 1$, i.e., a content can not be scheduled for download more than once.

To be useful, a content must be prefetched before the user checks his/her social media streams. Taking into account $p_{nav}(t)$, i.e., the likelihood that the user may actively navigate social media contents at each of the K slots, we can calculate an *expected prefetch scheduling benefit* for any scheduling $Z = \{z_{i,k}\}$ as:

$$\begin{aligned} Benefit(Z) = {} & p_{nav}(1) \times \textstyle\sum_{i=1}^{|I|} (B(i,1) \cdot y_i(1)) \\ & + (1 - p_{nav}(1)) p_{nav}(2) \times \textstyle\sum_{i=1}^{|I|} (B(i,2) \cdot y_i(2)) \\ & + \cdots + \textstyle\prod_{k=1}^{K-1} (1 - p_{nav}(k)) p_{nav}(K) \times \sum_{i=1}^{|I|} (B(i,K) \cdot y_i(K)), \end{aligned}$$

where each term on the right side of the equation specifics the expected prefetching benefit if the next time when the user navigates his/her social media streams is at the k^{th} slot. On the other hand, the *expected prefetch scheduling cost* for the prefetch scheduling Z is:

$$Cost(Z) = \sum_{i=1}^{|I|} \sum_{k=1}^{K} C(i,k) \cdot z_{i,k}.$$

Now we formally define the O^{2}SMPS problem as:

$$\begin{aligned}
\text{maximize} \quad & Benefit(Z) - Cost(Z) && && \text{(2a)}\\
\text{subject to} \quad & \textstyle\sum_{i=1}^{|I|} s_i \cdot z_{i,k} \leq T_{prefetch} \cdot bw(k) && k = 1, \ldots, K; && \text{(2b)}\\
& \textstyle\sum_{k=1}^{K} z_{i,k} \leq 1 && k = 1, \ldots, K; && \text{(2c)}\\
& z_{i,k} \in \{0, 1\} && i = 1, \ldots, |I|; k = 1, \ldots, K, && \text{(2d)}
\end{aligned}$$

where (2b) specifies that the total amount of data can be downloaded in a prefetching slot is constrained by the average download bandwidth in the slot.

The above problem can be reduced to a Generalized Assignment Problem (GAP) [23] that assigns $|I|$ items to K bins and the value of each item varies for different bins it puts in. The problem is known to be NP-hard, and its efficient approximation algorithms has been studied extensively in literature. In this work, we adopt the polynomial-time algorithm by Martello and Toth [22] that provides an approximate solution to the problem under the overall time complexity of $O(K|I|logK + |I|^2)$.

The algorithm has two phases. The first phase tries to provide a reasonably good assignment uses a measure of the *desirability* of assigning content i to slot t, say $g_{i,t}$. It iteratively considers all the unassigned contents, and picks the content with the maximum difference between the largest and second largest $g_{i,t}$ to get assigned first. The intuition is such content is most *critical* since failing to assign it into its best slot will negatively impact the overall performance most. We let $g_{i,t} = \frac{f_{i,t}}{s_i}$, where $f_{i,t}$ is the gain of assigning content i to slot t derived from F, and s_i is the size of the content. So $g_{i,t}$ is a measure of the *unit gain* of the content i in slot t. In the second phase, once all contents have been assigned, the solution is improved through local exchanges. Readers are referred to [22] for the algorithm details.

Forecasting Network Connectivities and User Activity The prefetcher relies on two predictions for each of the prefetching slots to make scheduling decisions: (a) a network prediction in terms of the connectivity probability distribution vector $\overline{p_{net}}(k) = [p_{wifi}(k), p_{cell}(k), (1 - p_{wifi}(k) - p_{cell}(k))]$ as well as the bandwidth vector $\overline{bw}(k) = [bw_{wifi}(k), bw_{cell}(k), 0]$; and (b) a user activeness prediction as the likelihood $p_{nav}(k)$ that the user might navigate his/her social media streams in the slot.

Because people are creatures of habit, many existing profiling and forecasting techniques (e.g., location-based or time series prediction) can be used by both prediction problems to achieve highly accurate predictions. A notable technique is BreadCrumbs [24], which is a location-based prediction scheme. It tracks the movement of the mobile device and utilizes a simple Markov model to generate connectivity forecasts. Their evaluation results indicate a very good accuracy. The technique requires the location information either from the device's GPS or techniques like Place Lab [15] where the device has to communicate with a remote server to extract its location information from the information of its current WiFi access point and cellular towers. To mitigate this constraint, we applied a another simple Markov model based technique when the location information is not generally available.

We use a time-dependent Markov model for forecasting. The technique is applied to both network connectivity and user activity forecast. To forecast network connectivity, the states of the Markov model are *WiFi*, *cellular* and *offline*. To forecast user activeness, the states are *active* and *inactive*. The transition matrix depends on the time of the day. Since the prefetch schedule is slotted, say N slots a day, we maintain the same number of transition matrix to capture the probability of a transition from a state at slot t to a state at slot $t+1$. For each state in the model and time boundary between slots, the prediction component updates the corresponding Markov transition matrix whenever the model is in the state and transitions to another or the time is moved to a new slot. The future predictions can then be made from the trained transition matrix. For example, let $A(k)$ be the transition matrix for transition from slot k to slot $k+1$, given the network connectivity at slot k as $\overline{p_{net}}(k)$, we can approximate the network connectivity $k+1$ and slot $k+2$ as $\overline{p_{net}}(k)A(k)$ and $\overline{p_{net}}(k)A(k)A(k+1)$ respectively. Meanwhile, the average bandwidth vector $\overline{bw}(k)$ can be easily derived for each time slot.

5 System and Application Implementation

As a proof-of-concept, we implemented O^{2}SM as a userspace Java library targeted at the Android platform. Our initial prototype supports prefetching of social media streams from Facebook. In the future, we will extend the system for Instagram and Twitter streams as well. Fig. 2 depicts the implemented system that includes two layers, the O^{2}SM middleware and *oFacebook* app.

The middleware layer runs as a service on Android, which contains two main threads, one to collect the meta-data for the social media stream and the other to prefetch media contents. Once the user logs into his/her Facebook account, collection of meta-data from newly uploaded contents is initiated. The prefetching thread implements the utility based prefetch algorithm discussed earlier - in principle, we can plug in alternate prefetch techniques (e.g., prefetch all etc.). The middleware can also fetch media contents on-demand when it receives requests from the oFacebook app.

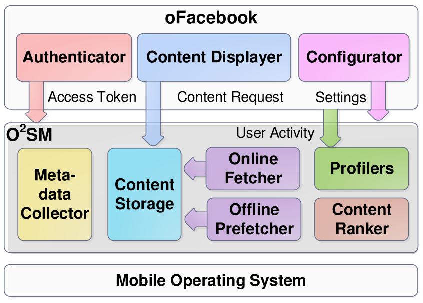
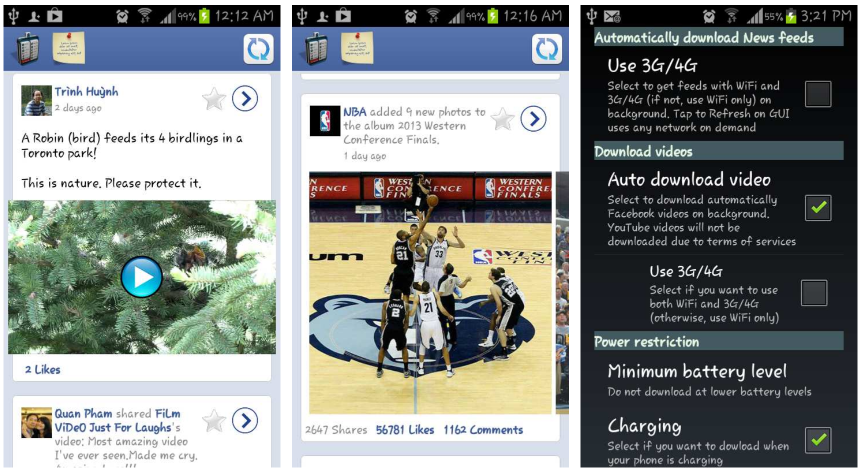


Fig. 2. The proposed system's implementation.

The oFacebook app (see Fig. 3) is implemented on top of the O^{2}SM middleware. The oFacebook app begins with the user logging into Facebook through its *Authenticator* component. Once the user is logged in and authenticated, the metadata is obtained, content ranked and downloaded, he/she can interact with the downloaded social media stream, e.g., view a photo, watch a video, see comments and likes. Content items are downloaded through the middleware either on-demand or via prefetching. The oFacebook app captures metrics(e.g., user clicks to view) and passes this to the middleware to learn how the user accesses the Facebook content - this will help the improve the ranking process with use. The oFacebook app also provides interfaces to change system settings (network and battery) through its *Configurator* component.

In our implementation, O^{2}SM stores the prefetched contents in local storage so that they can be accessed later via social media applications. The identifiers of the contents



12:12 AM

Trình Huỳnh
2 days ago

A Robin (bird) feeds its 4 birdlings in a Toronto park!

This is nature. Please protect it.

2 Likes

Quan Pham shared FiLm ViDeO Just For Laughs's video: Most amazing video I've ever seen.Made me cry.

12:16 AM

NBA added 9 new photos to the album 2013 Western Conference Finals.
1 day ago

2647 Shares 56781 Likes 1162 Comments

3:21 PM

Automatically download News feeds

Use 3G/4G
Select to get feeds with WiFi and 3G/4G (if not, use WiFi only) on background. Tap to Refresh on GUI uses any network on demand

Download videos

Auto download video ✓
Select to download automatically Facebook videos on background. YouTube videos will not be downloaded due to terms of services

Use 3G/4G
Select if you want to use both WiFi and 3G/4G (otherwise, use WiFi only)

Power restriction

Minimum battery level
Do not download at lower battery levels

Charging ✓
Select if you want to dowload when your phone is charging

(a) (b) (c)

Fig. 3. Snapshots of the oFacebook app: (a) show a video content, (b) show an album content, and (c) set configuration parameters. App link: http://www.ics.uci.edu/∼dsm/oFacebook.

are their original URLs; if the same content is referred by multiple social media streams (e.g., a YouTube video is posted to both the user's Facebook and Twitter account), only one copy is prefetched and stored locally. This is realized by implementing a *get(URL)* API that the SM app calls to access a specific content. If the content has been prefetched, the content data is returned to the SM app; otherwise, the middleware fetches the content using the on-demand fetcher module. All prefetched contents will be purged after a specific period (configurable by user) to save storage - the default value is one week. A *mark(URL)* API allows a user to permanently store a downloaded content, and a *unmark(URL)* API to cancel a previous mark.

6 Performance Evaluations

To gain better understanding of the performance of the proposed system and algorithms, we conducted a trace-driven evaluation. We distributed the oFacebook app to 10 participants located in America, Asia and Europe to run for 10 days from May 15th to 24th, 2013, and collected trace data to drive simulations to test the different algorithms. The users were asked to accept the terms of services agreeing to let us store and send logs for evaluation purposes. To secure the participants' data, we did not log any private content or person such as user names, message content, post descriptions and photo/video URLs. We represent Facebook users by hashed user ids and represent Facebook contents and multimedia objects (i.e. photos and videos) by hashed object ids using an one-way hashcode function to prevent the trace back to the original information. The participants use our system similarly to the standard mobile Facebook app: they install the system through an Android executable file (.apk), login to Facecbook with their Facebook account, and enjoy the downloaded Facebook newsfeed stream through a Facebook-similar navigation GUI. In the measurement version distributed in the user study, the ranker predicts a viewing probability of 1 for all posted content; the prefetching thread simply downloads all media contents - this allows us to measure user-content interaction behavior without bias. Some information collected in the collected log traces is listed:

- *Content*: information on social media contents such as created time, author id, number of likes and comments, multimedia file size. (There are 12,596 contents collected in 10 days for all users. The maximum number of contents for a single user is 3,919 while the minimum number is 130.)
- *user Activity*: the time and content id when users click to view contents. (A single participant clicks 693 times on average to view the downloaded contents. The maximum ratio of the number of views to the number of contents for a single user is 95% while the minimum ratio is 17%.)
- *Friend*: the ids of friends of users. (A single user has 310 friends on average. The maximum number of friends a user has is 503 while the minimum number is 75.)
- *System*: information on the current network (e.g., connected or not, WiFi or cellular networks, signal strength, and etc) and battery (e.g., charging or not, battery level, and etc).

6.1 Integrated System Evaluations

We have implemented a trace driven simulator in Java to drive the evaluation of the proposed system. The simulator implements a time slotted system, and runs simulations for each of the ten users for a simulated time of 10 days using their own trace data. The simulator reads (a) the newsfeed stream trace to generate content items that arrive into the system; (b) the network trace for the network condition at the simulated time; and (c) the user activity trace for the viewing activity at the simulated time. Moreover, the PowerTutor energy model is implemented in the simulator to evaluate energy consumption for content downloads. The contents derived from the trace data are ranked by the proposed ranking technique. See Section 6.2 for a more detailed evaluation of our ranking scheme.

Since the middleware is intended to support multiple social media applications at the same time, the content/data load is expected to be higher than that from the trace where only one social media source is considered. To gain a better understanding of the performance of each prefetching algorithm under a much higher data load, we have implemented a synthetic stream generator to provide synthetic social media stream input to the simulator. The synthetic stream generator uses a Poisson model to generate social media contents that arrives to the system. Each social media content has a type and size that are drawn randomly from the collected trace data. We consider a parameter r, “video ratio”, to control the probability of drawing a video over an image in each synthetic content. Moreover, to emulate the ranking on synthetic contents, the generator assigns a viewing probability to each item using a uniform-random distribution. To emulate actual user viewing behavior on the ranked and downloaded content, we label content as viewed, again, using a uniform random distribution. Consequently, about 50% of the content will be considered as viewed, this yields an accuracy of 75% for the emulated ranker (closely matches results from trace data).

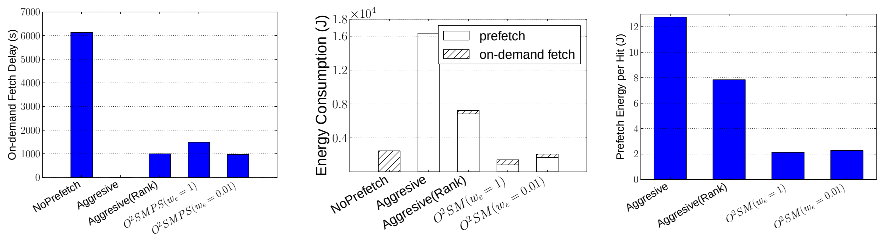
Besides the synthetic stream generator, we also implemented a synthetic network connectivity simulator to produce synthetic network connectivity. The purpose the synthetic simulation is to evaluate the system under different network environments that are not covered by the trace data. The network connectivity simulator is implemented using a Markov model with three states: "WiFi connectivity", "Cellular connectivity" and "no connectivity". State transitions occur every 15 minutes following the specified transition probability. The bandwidth of each state follows a Gaussian distribution, except for the "no connectivity" state whose bandwidth is always zero.

We consider the following performance metrics in our evaluation:

– *energy consumption*: which has 2 aspects: (a) prefetch energy consumption for contents that are prefetched and (b) on-demand fetch energy consumption for contents not prefetched, but requested.
– *prefetch energy per hit*: which is evaluated as the total prefetch energy over the number of prefetched contents being viewed by the user; since not all the prefetched contents are viewed, this metrics serves as an indication of both prefetch accuracy and prefetch energy efficiency.
– *on-demand fetch delay*: which is the downloading delay from on-demand fetching contents if they are not prefetched by the time they are viewed; since this metric aims to examine the latency that the user will experience for viewing unprefetched contents, we also considered the difference for fetching photos and videos in the simulator; For photo fetching, the delay is the latency for downloading the entire content. For video fetching, the delay is the latency for downloading the first 1 MegaByte considered as the playback buffer size.
– *cellular data consumption*: which is the amount of data downloaded through the 3G/4G interface.

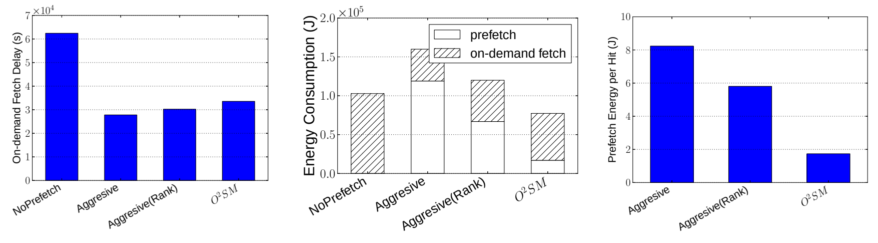
We compared our proposed prefetch scheduling algorithm with two baseline approaches. The"Aggressive" scheme periodically reads all feeds that arrive to the system but have not been downloaded, ordered from the newest to oldest, and sequentially downloads them whenever the network is available. The "Aggressive(Rank)" algorithm takes into account the content rank derived from the ranking algorithm. It periodically reads the most recent 100 feeds that have not been downloaded, but only downloads contents whose predicted viewing probability is larger than 50% (i.e. considered more likely to be viewed). Besides the baseline prefetching approaches, we also consider the conventional scenario of social media access where no prefetching is used in order to demonstrate the prefetching benefit.

Experimental Results Figure 4 reports the comparative performance of the O^{2}SM system with baseline schemes under purely trace based evaluations. We make several observations. All prefetching approaches are able to significantly improve a user's viewing experience by reducing the on-demand fetch delay for content viewing (Fig. 4a). The "'Aggressive"' approach improves the user's viewing experience to the most extent, by downloading every content possible. However, its prefetch energy consumption is significantly high (Fig. 4b). On the other hand, by taking into account the viewing prediction from the ranking technique, the "'Aggressive(Rank)"' and O^{2}SM algorithms provide much better energy efficiency by selectively downloading contents. Comparing the two we can see that while both algorithms provide a similar viewing experience improvement(Fig. 4a), the O^{2}SM algorithm uses only around 1/4 of the prefetch energy of the "'Aggressive(Rank)"' algorithm (Fig. 4b and Fig.4c) by intelligently scheduling the download when there are good connectivity.



(a) On-demand fetch delay (b) Energy Consumption (c) Prefetch Energy per Hit

Fig. 4. Evaluation on dataset.

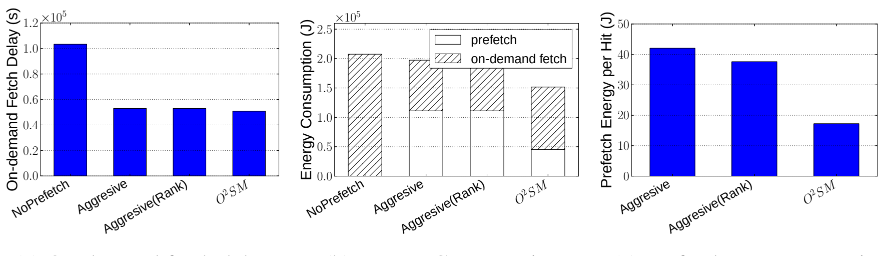


(a) On-demand fetch delay (b) Energy Consumption (c) Prefetch Energy per Hit

Fig. 5. Evaluation using synthetic contents where 50% are viewed.

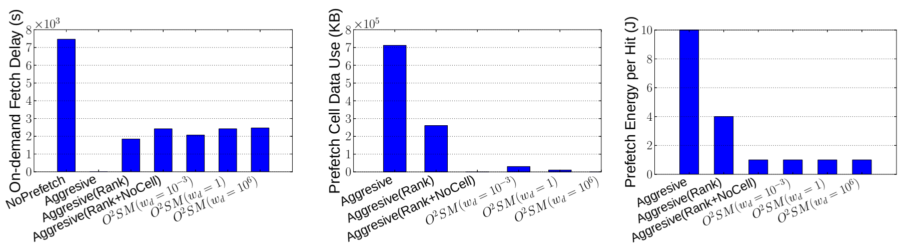
We also evaluated the O^{2}SM algorithm under different values of the w_e parameter, which indicates the significance of the energy evaluation in the cost/benefit analysis. We can see that with larger w_e, the algorithm is more conservative on prefetching. It achieves less energy use but results in less improved viewing performance because fewer contents are downloaded. One way to take advantage of the effect of the parameter setting is to let the system adaptively adjust the parameter value base on the current battery level. For example, we can adapt w_e to a lower (higher) value when the battery level is high(low)to enable more(less) aggressive prefetching and tradeoff viewing performance for energy conservation. We also tested the impact of the forecast range of network conditions and user activities on the performance of the O^{2}SM algorithm. We observe that when the forecast range is larger, the algorithm performs marginally better and achieves lower on-demand fetch delay and lower prefetch energy for a useful prefetch.

To gain a better understanding of the performance of each algorithm under high content/data loads especially when the network resources are not enough to ensure the prefetch of all contents, we evaluated them through synthetic simulation using the syn-



(a) On-demand fetch delay (b) Energy Consumption (c) Prefetch Energy per Hit

Fig. 6. Evaluation using synthetic contents where 100% are viewed.



(a) On-demand Fetch Delay (b) Cellular Data Plan Use (c) Prefetch Energy per Hit

Fig. 7. Evaluation using synthetic network connectivity.

thetic stream generator. The generator injects feeds into the system with Poisson mean rate of 1 feed per 5 minute, and the video ratio is 0.33, i.e. 1/3 of all the feeds generated are video feeds. We still use the trace data for network conditions.

Results in Fig. 5 show that under high content/data load, the "Aggressive" algorithm performs poorly; the high energy cost incurred (Fig. 5b and 6b) does not lead to a better viewing performance (Fig. 5a and 6a). This is because the aggressive scheme wastes network and energy resources on content that will not be viewed. On the other hand, O^{2}SM exhibits the best energy efficiency of all techniques: it also has the lowest total energy consumption and "prefetch energy per hit". We also evaluated the scenario where all contents will be viewed by the user. In this case, all content must be prefetched, and selective downloads based on rank will not help prefetch performance. The results are shown in Fig. 6. We observe that O^{2}SM can still improve prefetch energy performance by scheduling contents downloads when the mobile device has good network connectivity.

Since none of the users in our trace data have a 3G/4G data plan, to validate the system under cellular network connectivity we further evaluated the algorithms under synthetic network connectivity generated by the network connectivity simulator. The transition matrix for the network connectivity is randomly created, however, with the probability from any state to "cellular connectivity" larger than 50%. We still used the trace data for contents generated in the simulation. Fig. 7 shows the simulation results using synthetic network connectivity. We observe that by adjusting the w_d parameter, O^{2}SM adjusts the cellular data plan use to improve prefetch performance, while keeping a low prefetch energy cost.

6.2 Evaluations on the Content Ranker

We evaluate our ranking approach with Facebook data collected from the user study by employing 5-fold cross validation. Here, we randomly partition our data sets into 5 equal size subsets. Among the 5 subsets, 4 are used for training and the last subset is used for testing. This process is repeated 5 times, each time choosing one different set for testing. We report the ranking component's performance with two metrics that are widely used in designing recommendation systems: (i) the Receiver Operator Characteristic (ROC) curve and (ii) the area under the ROC curve (AUC). The ROC curve compares the number of contents that are viewed by user (i.e., positives) and correctly predicted to be viewed (i.e., true) with the number of contents that are not actually viewed (i.e., negatives) but incorrectly predicted (i.e., false). The ROC graph is plotted on two axes where the Y axis depicts true positive rate, which is equal to the number of correctly predicted positives divided by the number of positives, and the X axis shows false positive rate, which is the number of wrongly predicted negatives divided by the number of negatives. Intuitively, points in the upper left in the graph indicate better performance . The second metric, AUC, is a measure for the effectiveness of diagnostic tests; it is interpreted as the expected true positive rate, averaged over all false positive rates. An AUC that is closer to 1 indicates a higher accuracy. AUC is known to be a good metric to indicate accuracy for data set with skewed distributions.

Fig. 8a shows the AUCs for 10 participants. Our ranking component achieves high performance with the average accuracy of 71.7% (ranging from 81% to 62%). A random ranking in which contents are predicted randomly to be viewed or not be viewed (i.e., flip a coin) would yield 50% AUC. In Fig. 8b, we plot the ROC curves for three users 4, 5, and 7, whose AUC is the best, good and worst among the set of the participants respectively, to illustrate further the AUC results. The line of the circles in Fig. 8b represents the random ranking's performance. The ROC curve results for the users are consistent to the AUCs reported in Fig. 8a. For example, it is shown that the curve of user 4 with the highest AUC indeed dominates in Fig. 8b, and is much higher than the random ranking's line.

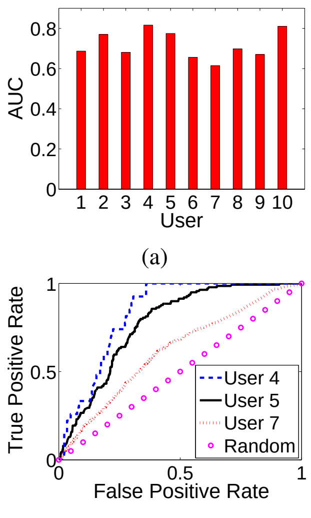


Fig. 8. Evaluations on the content ranker: (a) AUC for all users, (b) the ROC curve for three users.

We further show the running time of the training process. Since the whole data set is from our 10-day experiment, the data for the training process includes 8-day contents (we use the 5-fold cross validation). We use a DELL laptop with a dual core 1.8 Ghz CPU and 4 GB RAM running on Windows 7, and employ MATLAB (its logistic regression libraries are glmfit and glmval) to train and extract the learning model. The average running time to come up with a model is only 0.006 seconds while the maximum running time measured is 0.138 seconds. These results indicate that our ranking component is very efficient to mobile devices. Note that we do not need to train the model frequently, but run it once per day with a data set of the most recent 10 days to update the model with the current behaviors of the user.

We argue (and our results indicate) that the integrated system is inherently scalable for the following reasons. Since it is designed as a system that executes primarily on the user device (with little interaction between users for operation), we can scale to an arbitrarily large number of users (limited only by the network and OSN provider). Secondly, the ranking scheme, as illustrated is inherently low-overhead (avg. running time of 0.006 secs) and can scale well both in terms of the social network size and the number of content items that they upload - in general, the ranking overhead is minuscule compared with the download/prefetch overhead for the rich content. Finally, since the purpose of the system is to prefetch for future viewing, the sub-second latency has little to no impact on viewing delay.

7 Concluding Remarks

In this paper, we designed and developed an offline and online social media middleware system that prefetches media contents from online social networks for mobile users who suffers from interruptions from the Internet. Our system is equipped with two main components, media content ranker and prefetcher. Future work includes further research on cross social media networks, i.e., systems support multiple social networks. We also intend to explore the use of in-network resources, e.g., brokers/clouds, for storage and complex content ranking mechanisms. Eventually, the merge of social media/network and mobile computing will enable sharing and access of personalized content from multiple sources. This paper is an enabling step in that direction.

References

1. Canada-new media trend watch long-haul. `http://tinyurl.com/bv6p7mp`.
2. Facebook mobile app. `https://www.facebook.com/mobile/`.
3. FQL. `https://developers.facebook.com/docs/reference/fql/`.
4. Ndp report. `http://tinyurl.com/d3xq7dy`.
5. J. Ahn, P. Brusilovsky, J. Grady, D. He, and S. Syn. Open user profiles for adaptive news systems: Help or harm? In *16th International Conference on World Wide Web*, pages 11–20, 2007.
6. N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *9th Internet Measurement Conference*, pages 280–293, 2009.
7. F. Cacheda, V. Carneiro, D. Fernandez, and V. Formoso. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web*, 5, 2 2011.
8. X. Cheng and J. Liu. Nettube: Exploring social networks for peer-to-peer short video sharing. In *INFOCOM'09*, pages 1152 – 1160, 2009.
9. W. Chu and S. Park. Personalized recommendation on dynamic content using predictive bilinear models. In *18th International Conference on World Wide Web*, pages 691–700, 2009.
10. A. Devlic, P. Lungaro, P. Kamaraju, Z. Segall, and K. Tollmar. Energy consumption reduction via context-aware mobile video pre-fetching. In *IEEE International Symposium on Multimedia*, pages 261–265, 2012.
11. N. Gautam, H. Petander, and N. J. A comparison of the cost and energy efficiency of prefetching and streaming of mobile video. In *5th Workshop on Mobile Video*, pages 7–12, 2013.
12. M. Gemmis, P. Lops, G. Semeraro, and P. Basile. Integrating tags in a semantic content-based recommender. In *ACM Conference on Recommender Systems*, pages 163–170, 2008.
13. E. Gilbert. Predicting tie strength in a new medium. In *ACM Conference on Computer Supported Cooperative Work*, pages 1047–1056, 2012.
14. B. Higgins, J. Flinn, T. Giuli, B. Noble, C. Peplin, and D. Watson. Informed mobile prefetching. In *10th International Conference on Mobile Systems, Applications and Services*, pages 155–168, 2012.
15. A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, and F. Potter. Place Lab: Device positioning using radio beacons in the wild. In *3rd International Conference on Perasive Computing*, pages 116–133, 2005.
16. J. Leskovec, D. Huttenlocher, and J. Kleinberg. Predicting positive and negative links in online social networks. In *19th International Conference on World Wide Web*, pages 641–650, 2012.

17. Z. Li, H. Shen, H. Wang, G. Liu, and J. Li. Socialtube: P2P-assisted video sharing in online social networks. In *INFOCOM'12*, pages 2886–2890, 2012.
18. C. Lin, R. Weng, and R. Keerthi. Trust region newton method for large-scale logistic regression. *The Journal of Machine Learning Research*, 9:627–650, 6 2008.
19. D. Liu, G. Ye, C. Chen, S. Yan, and S. Chang. Hybrid social media network. In *20th ACM Internatioanl Conference on Multimedia*, pages 659–668, 2012.
20. H. Ma, I. King, and M. Lyu. Effective missing data prediction for collaborative filtering. In *30th International Conference on Research and Development in Information Retrieval*, pages 39–46, 2007.
21. H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *4th ACM International Conference on Web Search and Data Mining*, pages 287–296, 2011.
22. S. Martello and P. Toth. An algorithm for the generalized assignment problem. In *Operations Research*, 1981.
23. S. Martello and P. Toth. Generalized assignment problems. In *3rd International Symposium on Algorithms and Computation*, pages 351–369, 1992.
24. A. Nicholson and B. Noble. BreadCrumbs: Forecasting mobile connectivity. In *14th International Conference on Mobile Computing and Networking*, pages 46–57, 2008.
25. J. Noel, S. Sanner, K. Tran, P. Christen, L. Xie, E. Bonilla, E. Abbasnejad, and N. Penna. New objective functions for social collaborative filtering. In *21st International Conference on World Wide Web*, pages 859–868, 2012.
26. A. Rahmati and L. Zhong. Context-based network estimation for energy-efficient ubiquitous wireless connectivity. *IEEE Transaction on Mobile Computing*, 10:54–66, 1 2011.
27. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *10th International Conference on World Wide Web*, pages 285–295, 2001.
28. A. Schulman, V. Navda, R. Ramjee, N. Spring, P. Deshpande, C. Grunewald, K. Jain, and V. Padmanabhan. Bartendr: A practical approach to energy-aware cellular data scheduling. In *16th International Conference on Mobile Computing and Networking*, pages 85–96, 2010.
29. S. Traverso, K. Huguenin, I. Trestian, V. Erramilli, N. Laoutaris, and K. Papagiannaki. Tailgate: Handling long-tail content with a little help from friends. In *21st International Conference on World Wide Web*, pages 151–160, 2012.
30. C. Xu, C. Dale, and J. Liu. Statistics and social networking of youtube videos. In *16th International Workshop on Quality of Service*, pages 229–238, 2008.
31. T. Yan, D. Chu, D. Ganesan, A. Kansal, and J. Liu. Fast app launching for mobile devices using predictive user context. In *10th International Conference on Mobile Systems, Applications, and Services*, pages 113–126, 2012.
32. L. Zhang, B. Tiwana, R. Dick, Z. Qian, Z. Mao, Z. Wang, and L. Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, pages 105–114, 2010.