LOCAI: A Personalized Cache Mechanism for Location-Based Social Networks

Dimitrios Tomaras, Ioannis Boutsis, Vana Kalogeraki
Department of Informatics
Athens University of Economics and Business
Athens Greece
{tomaras,mpoutsis,vana}@aueb.gr

Dimitrios Gunopulos
Department of Informatics & Telecommunications
University of Athens
dg@di.uoa.gr

ABSTRACT

Recommending nearby Points of Interest (POI) has received growing interest in mobile location-based networks today, where users share content embedded with location information. In this work, we propose a novel caching framework to support personalised proactive caching for mobile location-based social networks. We propose "LOCAl", which uses a probabilistic approach in order to predict the POIs that users will access and retrieve the appropriate data objects that will fulfill user preferences. Our detailed experimental evaluation, using data from the Foursquare location-based social network, illustrates that LOCAl minimizes the user latency to retrieve the data objects they are interested in, is efficient and practical.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]

Keywords

Location based social networks, mobile caching

1. INTRODUCTION

Recently, we are witnessing a growing interest in exploiting location data in a wide variety of enterprise, consumer and public safety sectors. Examples of Location-Based Social Networks (LBSN) are found in a wide variety of domains from transport and carpooling systems that provide on-the-fly traffic updates or enable dynamic matching of drivers and passengers [3] to emergency response systems that utilize LBSN to deliver safety information and early warnings[1], to local search and discovery systems, such as Foursquare[1], that aim to recommend nearby Points-of-Interest (POI) to their users.

One of the most attractive services of location-based networks is recommending Points of Interest. POI Recommendation exploits the information that users share with their friends (*i.e.*, check-in into places of interest like museums, restaurants) and suggest POIs that users can visit, based on individual user preferences and feedback received from the community. POI recommendation is typically exploited by mobile users that issue location-aware queries, taking into account social, spatial, temporal and categorical aspects[8].

However, existing POI recommendation systems typically ignore the benefits of location-based caching. One important advantage is that caching can greatly reduce the latency and the corresponding cost of the data to be downloaded when users are required to use mobile networks (*e.g.*, 3G/4G) that could be very expensive in some cases (*e.g.*, for roaming users).

Selecting the data objects to be cached is a challenging problem as the storage on mobile devices is limited, especially for rich media such as images and videos. Thus, a location-based caching scheme should ensure that the data objects stored locally on a user's device provide benefit to the user in terms of utility. Although previous works recognize the benefit of recommending POIs[4, 8, 9] and approaches exist in the literature for caching data [2, 5], they focus on either recommending POIs or caching social content based on schedule, but they do not attempt to use caching to determine what data objects from a LBSN should be locally stored, which is the focus of our work. Our goal is to rigorously explore the benefit of caching on personalized data object recommendations in LBSN using memory constrained smartphones.

In this work we propose LOCAl, a personalized caching scheme for mobile location-based social networks. A key characteristic of LBSNs is that the same user request may have a different response for different user personal preferences. The key idea is to estimate the set of data objects that a user will likely request in the future, which complies with her personal preferences. Given a user's personal POI recommendations, we extract information regarding the users and cache the respective data objects beforehand using WiFi networks. Our insight is that a person's previous locations and preferences are good indicators of her interests and desired POIs. Thus, we focus on the following questions: *1) How can we effectively exploit the category preferences of a user to recommend a set of possible POIs to visit?* and *2) How can we determine a personalised set of data objects that a user will desire for these POIs, to proactively cache them on the user device?* Our experimental evaluation illustrates the accuracy and benefit of our proposed approach.

[1] http://foursquare.com

SIGSPATIAL'16, October 31-November 03, 2016, Burlingame, CA, USA

DOI: http://dx.doi.org/10.1145/2996913.2996981

2. SYSTEM MODEL AND PROBLEM DEFINITION

System Model. We assume a *Location-based Social Network (LBSN)* where mobile users are able to share social content that includes location-embedded information. We define as *Point of Interest (POI)* a distinct "venue", such as a restaurant or a museum, located at a specific geographical location. We denote each POI $p \in P$ with a tuple: $\langle (lat_p, long_p), cat_p, name_p, Data_p \rangle$, that includes the GPS location of the POI (lat_p, $long_p$), the category it belongs to (cat_p), its name ($name_p$) and a set of data objects which are related to the POI, such as reviews, photos, ratings, tips, etc; such information is typically exploited by users to decide whether to visit a venue.

Assume a set of users U of the LBSN. Sharing spatiotemporal presence of a user $u \in U$ in a POI p is denoted as a *check-in*, defined as $c_u^p \in C_u$, where C_u is a list that contains all check-ins of user u. A check-in c_u^p of user u at POI p is a tuple of the following form: $c_u^p =< u, p, timestamp >$, thus, it is characterized by the user, the POI and the current timestamp.

Each POI has a specific category cat_p, and thus, whenever a user u performs a check-in c_u^p at POI p we represent this dependency as $c_u^p \in cat_p$. We capture the category preferences of a user u, denoted as $pref_u^{cat}$, which is calculated as the fraction of the number of check-ins to the POIs of the specific category cat_p over the sum of check-ins in all categories visited by the user as follows:

$$pref_u^{cat_p} = \frac{\sum_{C_u} |1|, \forall c_u^p \in cat_p}{\sum_{C_u} |1|} \quad (1)$$

Finally, for each user we maintain a list of the objects that the user was interested to observe, for a specific POI p. This is denoted as Sel_u and contains a subset of the data objects $Data_p[obj]$ from all POIs that the user has visited. For instance a user in Foursquare may be interested in a specific set of images for a POI and the price range of the POI to determine whether she will visit it or not, while another user may be interested in the tips.

User personal devices in every major mobile operating system such as Android, iOS, Window Phone, etc., utilize a fraction of their local storage, which is denoted as *cache*, to maintain data that users may access. Caching has important advantages as (i) it minimizes the response time to display the requested content and (ii) reduces the total amount of requests in the mobile network. Data is stored in the cache, whenever users browse the internet via their smartphones or when mobile applications download data in the background to improve the user experience.

Problem Definition. Assume a geographical region R where a set of $N \subset U$ users equipped with mobile phones are located. Our goal is to dynamically predict which POIs a user is likely to visit and proactively cache the most interesting objects for the user from these POIs. Let $Pr_{p->p'}^u$ represent the probability that a user u will visit a POI p' based on her current POI p. Given this probability and the user previous selection behavior (Sel_u) we determine the probability that each individual object from POI p' might be requested by user u as $Pr(Data_{p'}[obj]|Pr_{p->p'}^u, Sel_u)$. Each data object is associated with a cost, which is determined by the size of the object in bytes, denoted as $size(Data_p[obj])$.

Our problem can now be expressed as the selection of the appropriate data objects to cache that will: (i) maximize the probability being selected by user u, given her previous transitions to POIs and her preferences to objects, and (ii) ensure that the total cost of the data items will not exceed the predefined cache size threshold, denoted as $Cache\_size$. Let variable x_i represent whether an object will be cached, and thus, our problem is formulated as follows:

$$\text{maximize} \sum_{i,p'} Pr(Data_{p'}[i]|Pr_{p->p'}^u, Sel_u) * x_i$$

$$\text{subject to} \sum_{i,p'} size(Data_{p'}[i]) < Cache\_size$$

$$x_i \in \{0, 1\}$$

By making the data available for the user before she actually requests it, the benefit is that: (i) the latency to receive the objects is reduced, and (ii) the energy consumption will be reduced since WiFi connections are much more energy efficient than mobile connections [6]. However, finding a feasible solution is a challenging problem. In the following section we show that the problem is NP-hard as it can be reduced to the Knapsack problem.

3. LOCAL APPROACH

For our approach, we define a *POI-to-POI Graph(PTPG)* $G = (V, E)$ which consists of a set of POIs V as nodes and the edges E. The edges denote the connections between POIs, providing us a comprehensible way to capture dependencies between different POIs. Our proposed algorithm consists of two phases: *A) the prediction phase* and *B) the caching recommendation phase.*

Prediction Phase: The prediction phase is responsible to create all necessary states based on user check-ins. In this phase, the graph model is built using historical data from all users. Whenever a user check-ins to a POI p' after a POI p, we define a transition $tr_{p->p'}$. Since the frequency of transitions is not affected by the uniqueness of individual users, the graph model is common for all users. In addition, the model is built only once and can be updated on a regular basis by fetching the appropriate information from a database server. Furthermore, by utilizing a common graph model for all users we are not obliged to rebuild the model whenever a user check-ins to a POI. In the prediction phase, a process that calculates the in-going and the out-going transitions from each POI p to any other POI p' is triggered.

Build States Process. While the majority of the existing works focus on predicting the transition of users between different POIs based on their category preferences or based on their current locations, our approach focuses on predicting the most probable cache state, i.e., the most probable set of data objects the users will desire at a future time, given their personal preferences. Such computations require a set of states to be initialized.

In our model, the POIs constitute the states of our model, thus, for every POI we create the appropriate state. Every POI keeps a queue of its data objects. These data objects are assigned the same transition probability $Pr_{p->p'}$. The out-going and the in-going connections between different state nodes represent the directed edges of our graph. We also utilize the number of occurrences of each out-going connection for the transition probability calculation. Each state keeps an internal list of all transitions, either in-going or out-going.

Caching Recommendation Phase: The Caching recommendation phase is responsible for the calculation of the probabilities and the construction of the final state of the cache. The caching recommendation is executed in two major steps. The first step is to calculate the appropriate transition probabilities in order to find the most probable POIs that users will visit. Then, by considering the popularity of each data object, we derive the final selection probability for each individual data object and retrieve the most appropriate ones, while also considering user preferences and cache size constraints.

Step 1. Transition Probabilities Calculation. Once the prediction phase is completed, the algorithm advances to the transition probabilities calculation. Given the check-in history and the current check-in of user, we first calculate the probability $Pr_{p\to p'}$ for two given POIs p and p'. In our system model, each data object $Data_{p'}[obj_i]$ has a specific field denoting the transition probability from POI p to POI p'. This field is assigned with the value of the computed probability $Pr_{p\to p'}$. First, we search in the graph model for the appopriate POIs p and p' which is executed in constant time by using a hashmap as index to state nodes. Once identified, the calculation of transition probability is performed by taking the number of occurrences of the transition from POI p to POI p', divided by the total number of transitions to any other POI p'' from POI p(which is derived by the size of the list that tracks the outgoing transitions). More formally:

$$Pr_{p\to p'} = \prod_{k} \frac{N_{p\to p'}}{\sum_{\forall p''} N_{p\to p''}} \tag{2}$$

User personalization is leveraged here. We select the top-K POIs by taking into consideration user's u category preference. For each $cat_p \in C = \{cat_p, cat_{p'}, cat_{p''}, ..., cat_{p''''}\}$ that user has performed a number of check-ins, we select the top-K POIs with high transition probabilities. In order to perform the transition probability calculation, we fetch all transitions from our data structure that begin from POI p and the destination POI p', which belongs to category $cat_{p'}$. For the POI selection, the category preference must be calculated. Each category cat_p of user u is quantified using the visit frequency fr_{cat_p}. Finally, we generate the transition probability from POI p to POI p', which will be common for all data objects. The POIs from each category are stored in a list denoted as CND, as input to the next step of the algorithm.

Step 2. Selection Probabilities Calculation. Upon the completion of the transition probability calculation process, we need to calculate the selection probability of every data object contained in the list CND returned from the transition probability calculation step.

Every data object in our approach has a specific variable $f_p[obj_i]$ that denotes its popularity among users. We have employed a queue of data objects using $f_p[obj_i]$ as the key for sorting every POI. By applying user u preference over textual or image information we filter the queue of objects l_p to contain objects of specific type (*e.g.,* image or text) or to provide a percentage for each of them (for example 70% images and 30% textual information).

Once the transition probabilities are calculated, our next step is to calculate the selection probability for the data objects given the personal preferences of the user. In order to select the final set of objects that will be cached at user smartphones, we select the data objects as the product of the transition probability $Pr_{p\to p'}$ with data object's popularity $f_p[obj_i]$. Thus, the final data object selection probability of object $Data_p[obj_i]$ can be derived as:

$$Pr_{total,Data_p[obj_i]} = \left(\prod_{k} \frac{N_{p\to p'}}{\sum_{i=1}^{|POI|} N_{p\to p_i}} \right) \cdot f_p[obj_i] \tag{3}$$

Our algorithm provides the option to choose between cost limitation and object limitation. Given the calculated probabilities and the option between cost minimization and object limitation, our greedy approach performs as follows: (i) Cost limitation: We sort the list according to object selection probabilities and we iterate over it, adding the cost of each object until the bound cost is reached. Then the objects are inserted in the queue D which is returned to users and sorted according to their selection probabilities. (ii) Object limitation: We sort the list according to object selection probabilities. We iterate over the list and we select a max number of objects given as bound. The data objects are inserted appropriately into the queue D, which is downloaded to users mobile phones. The majority of experiments was conducted with the object limitation option, since it provides a stricter environment for the object selection. Cache can be tuned to accommodate a greater number of multimedia information, whereas, when using object limitation, we restrict the number of POIs to predict and the number of data objects to select simultaneously.

4. EXPERIMENTAL EVALUATION

Experimental Setup. We have conducted our experiments using a dataset that includes check-in data in New York city collected from Foursquare from 12 April 2012 to 16 February 2013 [7]. The dataset contains 227428 check-ins and 1083 users, where the user identifier was anonymized for privacy reasons. Figures 1, 2, 3 and 4 illustrate an analysis of the dataset characteristics. As can be observed from figure 1, there is a small amount of users with a large number of check-ins, whereas the majority of the users have fewer than 500 check-ins during the considered time interval. Figure 2 shows that there exist specific categories which are highly preferred by users, such as Bars. Respectively, as can be observed from figure 3 there are approximately 1000 POIs that users tend to visit and check-in a lot, while the rest of the POIs have only a few check-ins. Finally, figure 4 presents that over almost 80% of our users check-in at 50 distinct POIs. As can be observed from the figure, only few users discover new places, whereas over 80% of users check-in to 1 up to 100 distinct POIs.

Evaluation. We conducted a set of experiments using our evaluation data to observe the benefits of our approach towards caching. For the purposes of our experimental evaluation, we have set up the following scenario. We build our model using all POIs from our dataset. However, we use only 60% of our dataset to train our model and the rest 40% for the validation process. For our experiments, the train dataset starts from 12 April 2012, up to the day that denotes the end of 60% percent of checkins (the actual day

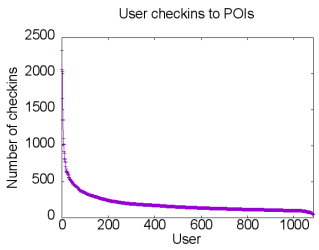

Figure 1: Checkins of users.

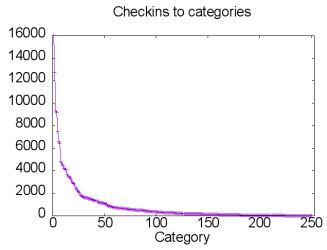

Figure 2: Checkins in categories.

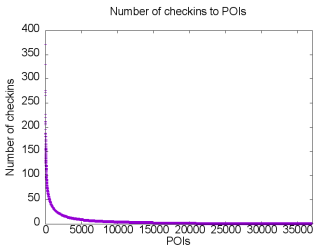

Figure 3: Checkins to Points of Interest.

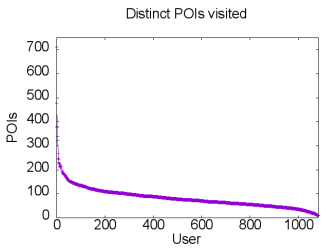

Figure 4: Checkins to distinct POIs.

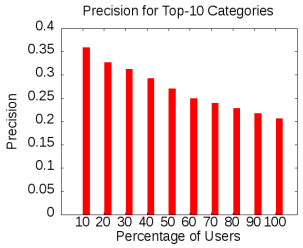

Figure 5: LOCAl precision for Top-10 user categories

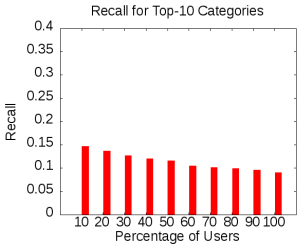

Figure 6: LOCAl recall for Top-10 user categories

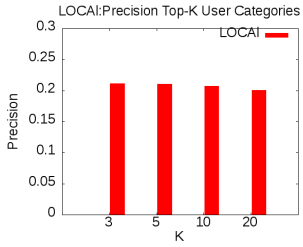

Figure 7: Top-K categories precision

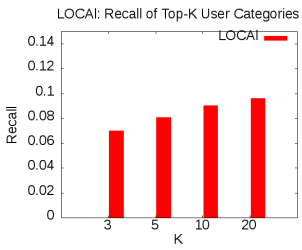

Figure 8: Top-K categories recall

is 18 July 2012).

In this set of experiments we evaluate the accuracy of LOCAl in terms of POI prediction. Figures 5,6 illustrate the precision and recall values to predict the next POI for the top-10 categories of each user where we vary the amount of the evaluated users. As we present, the precision is almost 35% for the 10% of the users and drops to 20% for the full set of users. The recall on the other hand varies between approximately 14% for the 10% of the users and is smaller than 10% for the full set of users. The reason that the precision and recall descent is because as we examine more users, the number of users for whom we cannot make any recommendations increases, thus, the mean value of precision and recall are lower. We also note, that, although the precision and recall values are low, they are comparable or higher from existing state-of-the-art recommendation approaches[4].

We also evaluate our approach as we vary the number of categories to choose for the prediction process from 3 to 20 for the full dataset. Despite the fact that precision remains approximately at the same levels, as shown in figure 7, the recall increases up to 10% for the top-20 categories, as illustrated in figure 8. This is due to the fact that for higher values of K, we select from more user categories, and thus, there are more possible POIs which are likely to be visited.

5. CONCLUSION

In this paper, we presented LOCAl, a novel caching framework which proactively caches sets of data objects that users will desire to access at POIs. Our experimental study with a real-world dataset illustrates that LOCAl can decrease the latency and the size of the data that need to be downloaded in a LBSN, and thus, improve the Quality of Experience for the users.

Acknowledgment

This research has been financed by the European Union through the FP7 ERC IDEAS 308019 NGHCS project and the H2020-ICT-688380 VaVeL project.

References

[1] Alexander Artikis et al. “Heterogeneous Stream Processing and Crowdsourcing for Urban Traffic Management.” In: *EDBT*. 2014, pp. 712–723.

[2] Ngoc Do et al. “Optimizing offline access to social network content on mobile devices”. In: *INFOCOM*. IEEE. Toronto, Canada, Apr. 2014, pp. 1950–1958.

[3] Xiaoyi Duan, Cheqing Jin, and Xiaoling Wang. “POP: A Passenger-Oriented Partners matching system”. In: *ICDEW*. IEEE. Seoul, South Korea, Apr. 2015, pp. 117–118.

[4] Xutao Li et al. “Rank-GeoFM: A ranking based geographical factorization method for point of interest recommendation”. In: *SIGIR*. ACM. Santiago, Chile, Aug. 2015, pp. 433–442.

[5] Carlos Lübbe et al. “DiSCO: A Distributed Semantic Cache Overlay for Location-based Services”. In: *MDM*. Vol. 1. IEEE. Lulea,Sweden, June 2011, pp. 17–26.

[6] Peng Shu et al. “eTime: energy-efficient transmission between cloud and mobile devices”. In: *INFOCOM*. IEEE. Turin, Italy, Apr. 2013, pp. 195–199.

[7] Dingqi Yang et al. “Modeling User Activity Preference by Leveraging User Spatial Temporal Characteristics in LBSNs”. In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45.1 (2015), pp. 129–142.

[8] Quan Yuan, Gao Cong, and Aixin Sun. “Graph-based point-of-interest recommendation with geographical and temporal influences”. In: *CIKM*. ACM. Shangai, China, Nov. 2014, pp. 659–668.

[9] Jia-Dong Zhang, Chi-Yin Chow, and Yanhua Li. “LORE: exploiting sequential influence for location recommendations”. In: *SIGSPATIAL*. ACM. 2014, pp. 103–112.