Mobile User Profile Acquisition Through Network Observables and Explicit User Queries

Nilton Bila
Dept. of Computer Science
University of Toronto
Toronto, Ontario M5S 3G4, Canada
nilton@cs.toronto.edu

Jin Cao and Robert Dinoff and Tin Kam Ho
and Richard Hull and Bharat Kumar and Paulo Santos
Bell Labs, Alcatel-Lucent
Murray Hill, NJ 07974, USA
{jincao,rdinoff,tkh,hull,bharat,paulo}@alcatel-lucent.com

***Abstract*—This paper describes a novel approach for gathering profile information about mobile phone users. The focus is on information that can be used to enhance targeting of advertisements. (The ads might be delivered into the mobile phones, or to other devices such as the user's IPTV.) Unlike previous approaches, we combine network observable data (primarily logs of cell towers visited) with specific queries to the user (*e.g.,* to ask what kinds of activities the user does in a given region that he frequents). The user might be willing to answer occasional queries of this sort through offers of service discounts, or to be able to receive more relevant ads.**

The paper focuses on two key aspects of our approach. The first concerns the statistical techniques used to determine information about regions visited, along with the frequency of visits, typical durations, and typical visit times. These techniques were developed based on logs of 6 users with mobile devices over a period of several months. The techniques address issues that arise when a given small region is serviced by multiple cell towers (in which case oscillations between cell towers can be confused with movement between regions). The second key aspect concerns optimizing the order in which queries are presented to users, in a context where different query answers have different values. (The values of answers might be influenced by the mix of advertising campaigns from which ads are to be matched against users.) Optimization is NP-complete in a relatively general context. We develop a polynomial time algorithm which yields optimal sequences for the case where the family of queries to be asked satisfies a tree-based property. This is extended to create a heuristic polynomial time algorithm for the general case.

I. Introduction

At Bell Labs, we have started an initiative called the Intuitive Network Applications (INA), that aims to learn user profile data through a combination of machine learning and human-factors directed user interaction, and use that data for various value added services [1]. In a current effort, we focus on the use of INA principles for advertising purposes, driven by analysis of a user's movement patterns (logs of location changes over time), which can be used for targeted location-aware advertising. In this paper, we focus on two key parts of the INA framework as applied to the advertising scenario: a location clustering algorithm that determines the regions that a user frequents, and a near-optimal query sequencing algorithm that determines a good order in which to pose queries, such that the value to the advertisers is maximized.

The bulk of machine learning (*e.g.,* learning frequented regions) is applied in a batch mode, though some of it could be done on a more online/real-time basis (*e.g.,* determining if the user is traveling out of his home *region*). In addition, explicit user queries are sent to the user on a periodic basis. Business models for explicit user querying already exist in the marketplace, including free or discounted service offerings from Sugar Mama [2], Blyk [3] and others.

The project described here is at an intermediate stage. We believe that we have a sound framework that can provide an end-to-end solution, and we have promising preliminary results in the areas of the overall architecture, a location clustering that can be used with essentially any cellular phone, and a solid theoretical understanding of query sequencing to maximize advertiser value. Key areas of future work include further validation and refinement of the location clustering, incorporation of Human Factors aspects into the query sequencing framework, and testing various components against various user populations.

Importantly, our framework for learning about a user through statistical learning and explicit queries is quite generic. In particular, our query sequencing approach permits the use of arbitrary database schemas based on a modified Entity-Relationship (ER) model to provide the structure for what needs to be learned.

The contributions of this paper span three broad themes: location determination, place learning and recognition, and profile acquisition. There is a body of related research in each of these topics, however, to the best of our knowledge we are the first to offer an end-to-end system design that integrates these three.

Organizationally, Section II presents an overview of INA, along with a motivating example that illustrates the kinds of profile information we are focusing on for the advertising scenario. This section also includes related references concerning location determination. Section III presents an overview of our location clustering algorithm that aims to determine the frequented regions visited by a user, and compares with related work on place learning and recognition. Section IV describes (near-)optimal query sequencing for explicit user interaction, and compares with related work on profile acquisition. Brief conclusions are presented in Section V.

II. Overview and Motivating Example

The INA framework built at Bell Labs broadly deals with the following aspects:

1. User profile: This stores information about a user. This information may be pre-provisioned, learned based on analysis of logs of user behavior, or explicitly provided by the user through queries posed to him over time. Each datum of information in the profile can also be associated with a value, which indicates the relative importance or usefulness of that datum to the mix of applications under consideration.

2. User logs: Logs of various user activities/behaviors (*e.g.,* calls, location changes) are stored at an INA server. These logs are generated as a side-effect of normal user interaction with the telecom network (*e.g.,* making/receiving calls, moving from one location to another). Note that the notion of user opt-in or consent for capturing and storing this information is outside the purview of INA.

3. Statistical analysis: Periodic analysis of available user logs is performed in order to generate a distilled summary of the user's habits and intentions. For example (as will be discussed in detail later on in the paper) analysis of location logs can be used to determine the *frequented* regions that a user visits.

4. User queries: Periodically, pre-created queries are posed to the user in order to obtain information that cannot be determined via statistical analysis alone. A query sequencing algorithm is used to determine the order in which to ask the queries, such that the value of the resultant user profile instance is maximized in the shortest sequence possible.

5. Realtime adviser: Various applications (*e.g.,* an ad-matching application) can query the user profile on demand. The real-time adviser performs additional filtering/interpretation of the user profile, before returning the desired results.

We now discuss these components in the context of a scenario which illustrates the challenges (and opportunities) of acquiring user profiles in a mobile environment. For ease of presentation, we focus on a simple user profile schema, key parts of which are shown in Figure 1. This schema stores the *regions* that a user frequents, the specific *locations* that a user visits within a given region, and the *activities* he performs at a given location (or broadly, in a given region). The location (activity) is associated with a location (activity) *type*, the corresponding types being hierarchically organized according to a specific ontology. Once such information has been partially (or completely) gathered for a particular user, it can then be used to provide him with targeted advertisements for products and services which are of interest to him (*e.g.,* coupons for a grocery store near his home).

The data model we use is based on the Entity-Relationship (ER) model. We assume that all relationships (shown as diamonds) are binary, and that attributes are on entities but not relationships. We generally blur the notion of an object identifier with the value of the scalar key value for that object. We extend the traditional ER model with *enumerated, subtyped domain* (*ESD*) entities. Each ESD entity ranges over a finite set of scalar values. ESD entities are shown as octagons, and

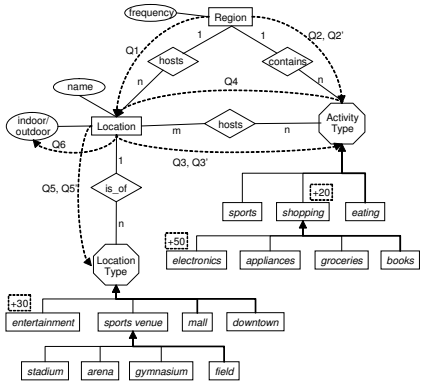


Fig. 1. Simplified user profile schema. The schema is annotated with a query graph which indicates the entities and attributes each query operates on. (A representative set of values for the `LocationType` and `ActivityType` entities is included for purposes of illustrations.)

the domain is shown as a set of rectangles, with the subtyping relationship indicated by a tree-based structure. ESD entity types differ from traditional subclass relationships, since ESD entity domains hold the actual names of types and sub-types, not the members of those types and sub-types.

There are two modes for populating this schema with information obtained about the users; firstly via automated analysis of user logs, and secondly via explicit queries to the user. We now discuss these two modes in detail.

Certain portions of the schema are obtained via statistical analysis of user behavior. In this paper, we focus on analyzing time-stamped logs of a user's location to determine the regions that the user frequents, and the times and durations during which he is in those regions. This can later be used for location-aware ad matching, among others.

Various techniques can be used to determine a user's location. GPS is commercially available, and offers reasonable accuracy. The challenge with GPS however, is that it is impractical for continuous location logging on mobile devices as it shortens the lifetime of the mobile device's battery significantly [4]. In addition, GPS is known to have poor coverage when used with devices carried in users' pockets, *e.g.,* an average of only 4.5% coverage in an urban environment has been reported [5].

Also popular in the research community is location sensing by having mobile devices listen to beacons of existing radio infrastructure such as WiFi access points or cell towers [5], [6], [7]. These techniques are critically dependent on having the geographical coordinates of the radio beacons being made available.

We focus on cell id monitoring since it can be applied to a much broader range of mobile phones, works indoors as well, and has low battery requirements. The downside is that the granularity of location determined is coarse-grained especially for suburban/rural environments where the cell towers could be far apart. As in other work, this can be mitigated to some extent if the mapping of cell towers to corresponding geographical coordinates is made available. Even in the absence of such information, statistical techniques can be applied to

reverse engineer the cell tower coordinates by a combination of learned information (*e.g.,* a user's home region) and user specified information (*e.g.,* his actual home address), and then performing bulk analysis on a large number of such users.

For the purpose of this paper, we assume a hosted service model without operator support. In other words, the advertising service (utilizing the INA framework) is accessible by multiple network operators, however, no operator provides the coordinates of the cell towers in their network. (Note that most of the challenges discussed in this paper apply even in the case where such information is available.) In addition, we do not consider the problem of reverse engineering the cell tower locations from user specified data. That problem is left as future work.

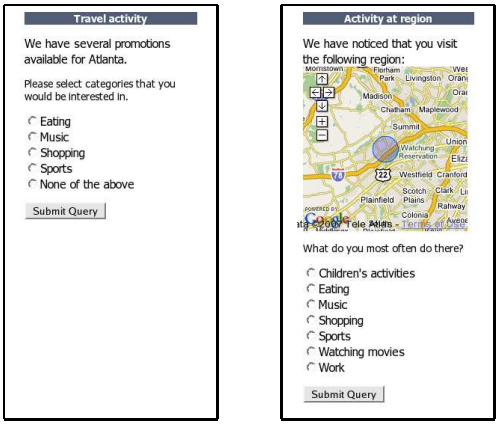
In the INA framework, a small client on the mobile device continuously monitors the cell ids the device is connected to. Periodically (*e.g.,* every 15 minutes), it sends this information to an INA logging server. Batch analysis (*e.g.,* once a week) is then performed on user logs to determine the regions they frequent. Internally, the schema represents a region as a cluster of cell ids visible at multiple visits to that region. An approximation of the region size could be made via techniques mentioned in the previous paragraph.

Since the determined regions may be coarse in nature, a user may actually be visiting multiple locations within a given region, and performing different activities at each location. If additional data (*e.g.,* user credit card statements) was made available to INA, it could be possible to analyze that information to determine the user's locations/activities for a given region. In the absence of such data feeds however, this information can only be obtained directly from the user. Hence, the second mode of obtaining information about a user is via explicit user queries.

We support the notion of query templates, which are essentially parameterized queries that can be posed to the user. Query templates aim to navigate the user profile schema, and fill in the data items that cannot be learned (queries can be used to confirm/validate learned information as well).

Figure 1 shows several query templates. Here `Q1` asks for `Locations` in a `Region`, `Q5` asks for the `LocationType` of a `Location`, and `Q5′` can ask for the sub-`LocationTypes` of a `LocationType`. Query `Q6` asks if `Location` is indoors, outdoors, or both (*e.g.,* a shopping mall or downtown area that includes both buildings and outdoor connections between them). Query `Q2` can ask for `ActivityTypes` that a user does in a `Region`, and `Q2′` can ask for the sub-`ActivityTypes` of an `ActivityType` that are performed in the `Region`. Query `Q3` asks for `ActivityTypes` associated with a `Location`, `Q3′` asks for sub-`ActivityTypes` associated with a `Location`, and `Q4` asks for the inverse relationships.

Visual representations of two sample query templates are presented in Figure 2 (this figure assumes that the queries are asked via a browser on the user's mobile device). The query in Figure 2(a) tries to determine the activities that a user might be interested in if he is found to be traveling. This query is



(a) Travel activity template (b) Region activity template

Fig. 2. Visual representations of two sample query templates

simple in nature; more generally, we are interested in filling out the user profile with some degree of detail. The second query (Figure 2(b)) aims to determines the high-level activities performed at a given region, the set of possible activities being derived from an ontology maintained by INA. This query template is invoked for any new frequented region that has been learned via location analysis, and for which no activity information is available.

Not all queries are created equal. Different advertising campaigns will be interested in different characteristics of users, and so the mix of campaigns will influence the value of different kinds of information (*e.g.,* whether a user does shopping for electronics within a region or not). Hence, we associate a *value* to each query template (or to the data it obtains during a specific *instantiation*) that can be posed to users. Some representative values are shown in dashed squares in Figure 1. A key contribution of this paper is to determine in which order to pose queries to a user, so that maximum value can be obtained in the fewest number of queries. We describe this query sequencing problem in greater detail in Section IV.

There is also an associated cost of queries. Querying users is intrusive, as it requires that users perform actions unrelated to their current activity. Moreover, there could be certain portions of the schema that a given user is sensitive about (*e.g.,* specific location information), and hence will be unwilling to answer queries for that information. We do not focus on these aspects in this paper.

III. Location Clustering

We investigate statistical learning techniques for maximally exploiting the information available from some minimum network observables. Specifically, we describe a two-step approach to identify frequented regions using time-stamped logs of cell tower IDs. Our method first segments the log into stationary and transit periods. Then it examines the stationary segments to identify a cluster of unique cell IDs. Finally, it filters and merges the clusters to generate candidates of frequented regions.

Our approach builds on [8], where the primary focus is to extract journeys from logs of cell IDs. While we look for

the transit periods first, our focus is on inferring the regions afterwords, similar to Ashbrook et. al. [9] and Marmasse et. al. [10], where stationary regions are inferred by the loss of GPS satellite signal. It is different from [6] where stationary points are first detected using a duration criterion (stable signal strength for 3 minutes). Also worth mentioning is that our focus here is different from Google's My Location mapping service [26], where the primary goal is to inform a user his/her location rather than learning the location-related behavior.

Our method places more emphasis and confidence on trainable statistical discrimination between transit and stationary periods, which makes it more adaptable to different cell tower environment. Training of our method for a given cell tower environment requires only the truth of at least one user's transit status for a limited period. This is less expensive than requiring detailed measurements of signal strength over a large area, as in Hightower et. al.'s BeaconPrint [11] (using WiFi and GSM radio fingerprint) and Rekimoto et. al's PlaceEngine [4] (using WiFi fingerprints), or careful tuning of distance parameters as in [12] and [13].

In refining the cell ID clusters, we use as a similarity metric the proportion of the user's time spent in common cells shared by distinct clusters. This is different from Kang et. al.'s temporal distance clustering [12]. We merge candidate regions as in Ashbrook's attempt to construct a Markov model to predict where users may go next. We now describe the method in detail.

A. Change Point Detection

A major challenge in segmenting the logs is to distinguish changes in cell IDs that are caused by the user's genuine movement from those caused by the system's nondeterministic choice of which cell a device is locked to. The nondeterministic lock is affected by the instantaneous radio environment, and it results in several nearby cell towers registered almost simultaneously in the log, recorded as rapid oscillations between two or more different cell IDs. We resolve this by a trainable decision procedure on whether the user is in transit or stationary.

A time-stamped trace of cell tower IDs is first converted to a time series of cell IDs with uniform temporal step sizes using simple interpolation. Denote this time series with n samples by (t_1, c_1), (t_2, c_2), ..., (t_n, c_n). For each time step t_i, a statistic h_i is defined as follows:

$$h_i = \begin{cases} 0 & \text{if } c_i = c_{i-1} \\ 1 & \text{if } c_i \neq c_{i-1} \\ -1 & \text{if } c_i \neq c_{i-1} \text{ and} \\ & c_i \text{ appeared within the previous 30 minutes} \end{cases}$$

Training of our algorithm can be done by one or multiple users. During training, a user is asked to mark the time when he is in transit. This marking distinguishes the training entries into two classes "$in\ transit$" or "$stationary$". The frequencies of entries of each class with different values for h are used to estimate the conditional probabilities $P(h = 0|in\ transit)$, $P(h = 1|in\ transit)$, $P(h = -1|in\ transit)$, and their counterparts in $P(h|stationary)$ for all values of h. Obviously, these probabilities depend on the travel speed relative to the cell sizes, and the oscillatory pattern of the cell IDs in the concerned geographical area. For example, if the travel speed is low relative to cell sizes, then the ratio of new cells $h = 1$ during transit will be low. For training purpose, we assume that the typical travel speed is more than 20 miles per hour, and we compute these probabilities for the worst case scenarios, *i.e.,* geographical areas where the cells are sparse. For high-speed travel, or for geographical areas where the cells are dense (like cities), even though these trained probabilities may not be accurate, they still suffice for the proposed change detection methodology below for automatically identifying the stationary and transit periods, albeit sub-optimally.

Change points in the time series are detected by a method known as "CUMSUM" (cumulative sum control charts)[14]. We use two CUMSUM charts S and D to detect the start and end points of a transit period, respectively. The algorithm alternates computation between the two charts, starting by turning on S with an initial value $S_i = 0$, and turning off D. Iterate on the time step i:

1) Compute the log-likelihood ratio statistic
$$l_i = \log \frac{P(h = h_i|in\ transit)}{P(h = h_i|stationary)}.$$
2) If S is on, compute the cumulative sum for the starting point
$$S_i = \max(S_{i-1} + l_i, 0).$$
If S_i exceeds a threshold T_{start}, signal that the user is in transit. Find the previous closest point t_{start} such that $S_i = 0$ and mark this the start of the transit period. Proceed to find the end of the transit period. Do this by turning off S, turning on D, and initializing D_i to 0.
3) If D is on, compute the cumulative sum for the ending point
$$D_i = \max(D_{i-1} - l_i, 0).$$
If D_i exceeds a threshold T_{end}, find the previous closest point t_{end} such that $D_i = 0$, and mark this the end of the transit period. Continue to look for the next starting point. Do this by turning off D, turning on S, and initializing S_i to 0.

The intuition behind the algorithm is as follows. With h_i at a specific value, we can have $P(h = h_i|in\ transit)$ larger than, equal to, or smaller than $P(h = h_i|stationary)$. Correspondingly, the log-likelihood ratio l_i will be positive, zero, or negative. S_i increases with the positive values. It exceeds the chosen threshold when the joint likelihood of the user being in transit in the previous time steps has been substantially *larger* than the joint likelihood that he is stationary. Similarly, D_i increases with the negative values of l_i, and its passing of a chosen threshold signals that the joint likelihood of the user being in transit is substantially *smaller* than that he is stationary. Figure 3 illustrates the operation of the S and D charts. The threshold values of T_{start} and T_{end} can be set by computing the average run length values (ARL) [14]. In our experiment, we set the thresholds somewhat lower since we do not want to miss any transits, but a mis-identified transit can usually be remedied by the cluster filtering step described in the following.

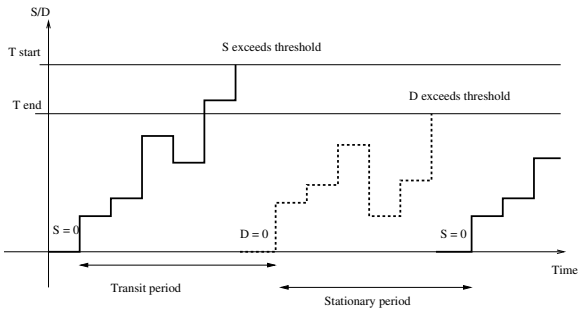


Fig. 3. Illustration of the CUMSUM method. The solid lines represents the S charts for detecting the starting point of the transit period. The dashed line is a D chart for detecting the ending point. Tracking alternates between S and D, switching when the active chart crosses the predesignated threshold.

B. Cluster Filtering and Merging

Cluster Filtering. The change point detection procedure results in a segmentation of the time series into alternating transit and stationary periods. Cells associated with the stationary periods are collected into clusters. Because of oscillations and imperfections in training, dubious transitions may result from the change point detection procedure. This is further guarded by applying two filtering criteria:

1) Duration: the duration of the transit has to be large than a threshold T_{dur} = 5 minutes. Transits of a shorter duration are often found to be oscillations.
2) Common Cells: compare the cells visited before and after the transit period. If there are common cells from the two sides, calculate the proportion of time that was spent in those common cells (p_{before}, p_{after}). If $\max(p_{before}, p_{after})$ is larger than a threshold T_{common}, discard the transit period in between.

To identify the stable clusters that do not straddle some other transition period, we check each cluster against each pair of clusters separated by other transition periods. If a cluster C contains cells included in two clusters before and after another transit period, label C as an unstable cluster and exclude it from the subsequent analysis. In doing so, we place much confidence in the transition periods detected by the CUMSUM procedure, but allow for the possibility of missed transitions. Only those clusters that never straddle any detected transitions are considered stable. The unstable clusters are to be broken apart using the knowledge of the stable clusters, and this can be done repeatedly.

Cluster Merging. Finally, as an additional protection against splits due to the nondeterministic lock, a hierarchical clustering procedure is applied to merge the stable clusters into frequented regions. Starting with individual cell clusters, the procedure recursively merges them into larger clusters, using a similarity metric defined to be the maximum proportion of time spent in any common cells between two candidate clusters. This is the same as the "Common Cells" criterion used in cluster filtering. The cluster tree is cut at a height H, which can be preset using common sense. In our experiment, we used $H = 20\%$. The clusters active at the cut height are identified as stable regions.

C. Examples and Plots

The proposed approach is tested in an experiment that is participated by six volunteers. Each user carries a smart mobile phone (HP iPAQ) that subscribes to a GSM service. Custom software logs the cell tower ID the phone is locked to at each pre-designated time step. Data were collected over several months. In addition, some participants entered their activity context and named the locations they visited for any significant period, and noted the fact that they were in transit. The truth data on transit status were used to train the change point detection algorithm.

Figure 4 shows the tree of stable clusters extracted from a user's log for two months using the proposed method. Figure 5 shows the spatial distribution of these clusters that is approximated using a multi-dimensional scaling procedure known as Sammon's nonlinear mapping[15]. The technique computes a layout of points in a plane, such that the planar distance between each pair of points is proportional to their distance given as input. In this application, the input distances are those used in producing the cluster tree (*i.e.,* the minimum fraction of time the user spent in the cells unique to each cluster). Clusters that have many common cell IDs thus tend to have small distance in between. Truth data from the user confirmed that the two dense regions inferred via the clustering algorithm contain his home and work locations, respectively.

IV. An Algorithm for Sequencing Queries

This section describes results and algorithms for finding optimal and near-optimal ways to sequence the queries that a system can pose to end-users, in order to fill in the non-statistical portions of that user's profile data. An overarching goal of these algorithms is to obtain as much data as possible from the user, with a preference for obtaining the high-value data as quickly as possible. A variety of non-technical factors can impact the effectiveness of an algorithm in achieving this goal, *e.g.,* the user's willingness to answer queries, or the user's ability to correctly understand the queries being asked. In this section we focus primarily on the technical aspects of choosing optimized query sequences.

Before delving too deep into asking privacy-invading and possibly annoying questions that users might just reject outright, one must ask whether users are even likely to answer such questions. We surveyed and interviewed users, and were encouraged by the results that suggest that, when presented with a useful service that can only be enabled by sharing personal information, a large enough fraction of users are willing to share such information in order to get the service. For example, in a 2007 study in the U.S and Hong Kong [25] we found that 94% of users who would like to be alerted of a transportation delay and alternate routes and believe that such a service could be provided, would not mind sharing their daily schedule and/or transportation habits. That same study, covering 236 respondents, had encouraging results also in other domains, spanning a broad range of applications from restaurant recommendations to library book reminders.

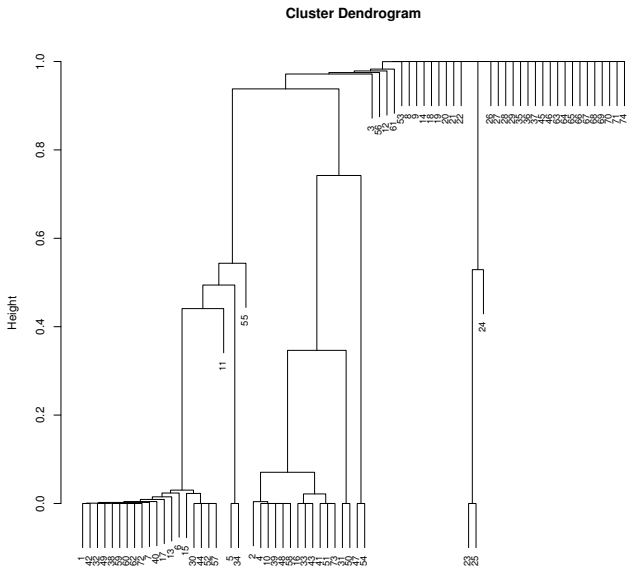


Fig. 4. Cluster tree resulting from hierarchical merging of the stable cell clusters.

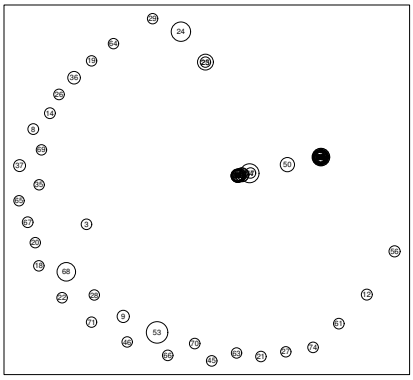
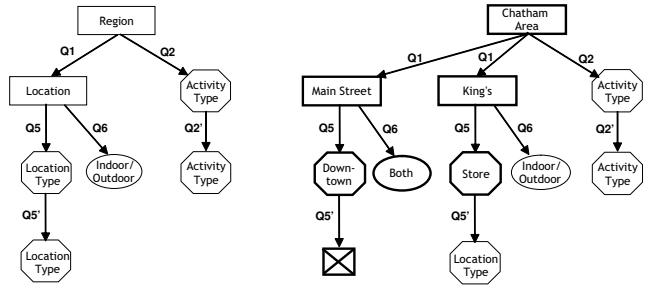


Fig. 5. Spatial distribution of the cell clusters approximated by multi-dimensional scaling using Sammon's mapping. The size of the circle representing each cluster is proportional to the time this user spent in the cluster.

Encouraged by these results, we decided to proceed with complementing user profile acquisition using direct user queries.

User profile acquisition is a topic of active research in the user modeling community (for surveys, see [16], [17]). Citations [18], [19], [20] seek to automate acquisition of user knowledge through inference (not explicit user queries). Stegmann [21] and Kass [22] acquire user profiles explicitly by means of a dialog with the user, and can dynamically add to the database schema as new kinds of information are obtained. The CALO (Cognitive Assistant that Learns and Organizes) project, similar to INA, has aimed to combine machine learning with explicit user input. For example, the PLIANT (Preference Learning through Interactive Advisable Nonintrusive Training) system [23] combines current user input with past history of his scheduling preferences, in order to propose an acceptable schedule for a given meeting. In contrast with the approach presented here, in all of the previous work the data acquisition does not take into account differing



(a) Abstract query tree (b) Hybrid query tree

Fig. 6. Abstract and hybrid query trees, assuming queries Q1, Q2, Q2', Q5, Q5', and Q6

values for different kinds of data. Further, our approach is relatievly generic, permitting use of arbitrary database schemas based on a modified ER model.

The algorithms presented here are fundamentally greedy algorithms, that incorporate the values associated with data, and probabilities concerning the anticipated characteristics of a user's profile data and willingness to answer. Due to space limitations, the section focuses mainly on the intuitions behind the algorithms, and does not provide detailed definitions for the concepts used, nor does it provide proofs of the correctness of the algorithms. These and other details may be found in [24].

Subsection IV-A describes our general framework, and develops an informative algorithm for a family of very specialized situations. Subsection IV-B generalizes that algorithm, and Subsection IV-C discusses further generalizations.

A. An algorithm for instantiated tree graphs

We begin by examining a very specialized case. Specifically, we focus now on (i) families of query templates that have a tree-based structure, and (ii) a single user.

For the remainder of this and the next subsection, we shall focus on a restricted, somewhat artificial variation on the schema of Figure 1. Specifically, we consider only the query templates `Q1`, `Q2`, `Q2'`, `Q5`, `Q5'`, `Q6`. Further, in order to illustrate some key intuitions, we assume here that there is no relationship between the `Location`s and `LocationType`s that a user visits and the `ActivityType`s that she performs. In Subsection IV-C we return to the general case, where this data is interdependent.

Figure 6(a) shows the *abstract query tree* corresponding to the family `Q1`, `Q2`, `Q2'`, `Q5`, `Q5'`, `Q6`. of query templates. (The construction of these trees is enabled by the structure of query templates. In general, we would work with a forest.) Here `Region` acts as the root, because this entity can be "seeded" by the statistical algorithms. (In this scenario, the system is not permitted to directly ask the user for `ActivityType`s that he performs, but rather, such questions must be related to specific `Region`s that the user visits.)

Conceptually, the user's answers to queries are recorded by modifying the abstract query tree, as suggested in Figure 6(b), which shows a *hybrid* query tree, and Figure 7, which shows a fully *instantiated query* tree. (Instantiated values are shown in icons with thick borders.) In the latter tree,

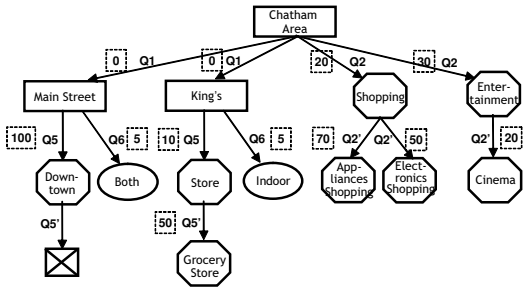


Fig. 7. Instantiated tree query with values, assuming Q1, Q2, Q2', Q5, Q5', and Q6

the statistical learning determined that the user frequents the "Chatham Area" region, and the user querying has determined two locations in that region that she goes to ("Main Street" and "King's"), and also two high-level activity types that she performs in the Chatham Area, and also sub-activity-types for those. The numbers in dashed rectangles in Figure 7 show a possible value assignment for information acquired about a hypothetical user.

We consider now the following question: Given complete knowledge of a user's profile, is there a polynomial time algorithm for sequencing queries to that user, in order to maximize the value of the information obtained? More formally, a *query sequencing function* is a function $f : \mathbf{N}^+ \rightarrow$ the set of instantiated query templates, which is defined for all positive integers, or for some initial sequence of the positive integers. We consider only query sequencing functions f which are *valid* for a user u, in the sense that if an instantiated query template is asked, then either it is based on a *seed value* (provided as the root value for some query tree) or it is based on a value already obtained by some previous query.

Given a query sequencing function f, and a fixed user u, we associate the *value function*, denoted $val[f, u](\cdot)$ or simply $val[f](\cdot)$ if u is understood from the context, defined so that $val[f](i)$ is the sum of the values of all the query answers to $f(1), \ldots, f(i)$. We are also interested in the *cumulative value* that f is providing to the advertisers, taking into account that once a query q is answered, then the answer provides value for every time interval after that.

We shall assume in the following that users always provide correct and complete answers to the queries asked of them. For example, when the user of Figure 7 is asked query `Q2` for the Chatham Area, she will respond with both `Shopping` and `Entertainment`, yielding a value of 50 points. A subsequent query of `Q2'` on `Shopping` will yield another 120 points of value. In contrast, asking `Q1` on Chatham Area yields two `Locations`, but no value.

An optimal query sequencing function for this user will be as follows.

$f(1)$	=	`Q2`(Chatham Area)	$f(6)$	=	`Q5'`(Store)
$f(2)$	=	`Q2'`(Shopping)	$f(7)$	=	`Q2'`(Entertainment)
$f(3)$	=	`Q1`(Chatham Area)	$f(8)$	=	`Q6`(Main Street)
$f(4)$	=	`Q5`(Main Street)	$f(9)$	=	`Q6`(King's)
$f(5)$	=	`Q5`(King's)			

We note that this is not based on a purely greedy algorithm; if it were, then $f(7)$, which yields 20 points, would be performed before $f(3)$, which yields 0 points. However, $f(3)$ allows $f(4)$ which brings 100 points. A key factor is the *slope* of a sequence of queries (or more correctly, of $val[f](\cdot)$), *e.g.,* $f(3); f(4)$ has a slope of 50, whereas $f(7)$ has a slope of only 20. Notions such as height and area under a sequencing function f are defined analogously.

A valid sequence of queries is *succinct* if there is no proper subsequence that is valid and has higher slope. The optimal algorithm for sequencing queries when there is full knowledge of the user is shown in Figure 8. This has running time which is polynomial in the the size of the user's profile database. With some straightforward optimizations, it can be modified so that each repeated step has running time linear in the number of trees plus the maximum size of an instantiated query tree.

Algorithm: Instantiated-Tree-Based
Input: Schema S, tree-based query template set $\mathcal{Q}$ value function v, user u
Output: Query sequencing function f valid for u
begin
  Construct the instantiated forest F for u
  Set $j := 0$
  Mark the root of each tree in F as "known"
  repeat until all nodes in F are marked as "known"
    For each tree T in F, identify all permitted succinct sequences
    Choose a succinct sequence s from F such that s has maximum slope and smallest size among all chosen succinct sequences
    Extend f for $j+1, \ldots, j+|s|$ so that it asks the queries listed in s
    Mark each node touched by s as "known"
    Set $j := j + |s|$
  end repeat
end

Fig. 8. Algorithm for tree-based schema with fore-knowledge

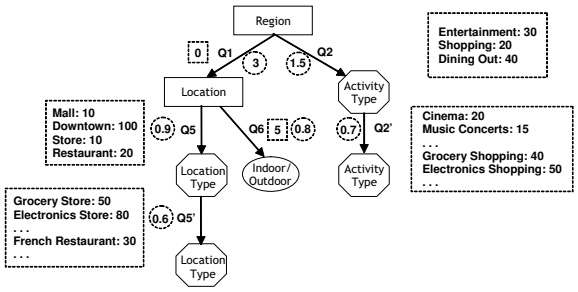


Fig. 9. Abstract query tree with schema-level values and expected counts, assuming Q1, Q2, Q2', Q5, Q5', and Q6

B. Algorithms for Generic Tree-Based Query Graphs

Using the intuitions developed in the previous subsection, we now describe a general algorithm for families of query templates that are tree-based, *i.e.,* whose graph corresponds to a forest of trees. As input to the algorithm we are given a generic value function, as illustrated in Figure 9. Although not considered in detail here, we assume that these values are determined from the mix of advertising campaigns that are to be supported. The value assignments may be at the level of schema components, may take into account the concrete domain element in an ESD entity (*e.g.,* knowledge that a user goes to a `Downtown` area may be highly valued), and may be relativized to seeds (*e.g.,* that information about user behavior in a certain `Region` has high value).

The algorithm also needs a statistical characterization of the profile databases of the user population. For example, how

Algorithm: Generic-Tree-Based
Input: Schema S, tree-based query template set $\mathcal{Q}$, value function v, expected count function δ, user u
Output: Query sequencing function f for u
begin
  Initialization:
    Construct the unary, seeded forest F for $\mathcal{Q}$,
    For each node n in F, find maximally preferred succinct subsequence $s(n)$
  Execution:
    Set $j := 0$
    repeat until all trees in F are fully instantiated
      Find a maximally preferred succinct sequence s that starts with a permitted node in F
      Set $f(j+1)$ to be the query at start of s
      Execute this first query in s
      Modify the appropriate tree T in F to reflect the value(s) obtained from answer. (Note that this may introduce new permitted nodes.)
      Set $j := j + 1$
  end repeat
end

Fig. 10. Algorithm for tree-based schema for generic case

many `Locations` are typically associated with a `Region`, etc. As a simple form of statistical characterization, we assign for each query template an *expected count* of answers. (We do not use a probability, since some query templates are asking about 1:n or m:n relationships.) Representative expected counts are shown in Figure 9 in dashed circles. We assume these counts to be statistically independent.

In this setting, there are several parameters of interest for a (valid, succinct) sequence s of queries: $\sigma(s)$ is the expected slope of s; $\eta(s)$ is the expected height of s; $\rho(s)$ is the expected area/length of s; and $\lambda_1(s), \ldots, \lambda_n(s)$, where λ_i is the probability of the instantiated version of s having length exactly i. Although space limitations prevent a full discussion, in general, we say that s is *preferred* over s' if $\eta(s)/(\Sigma_i(i \cdot \lambda_i(s))) \geq \eta(s')/(\Sigma_i(i \cdot \lambda_i(s')))$, and if these are equal, then if $\rho(s) \geq \rho(s')$.

Figure 10 shows the general algorithm for tree-based families of query templates. This will yield query sequencing functions with maximum expected utility. The basic idea is to mimic the behavior of Algorithm Instantiated-Tree-Based. However, as we gain knowledge of the user, we may opportunistically explore query templates, to take advantage of certain values already obtained. In the algorithm, a node is *permitted* if it is instantiated and it has at least one non-instantiated child. With suitable book-keeping, each step of this algorithm is again linear in the number of hybrid query trees that arise (*i.e.,* the number of seeds) plus the size of the maximum abstract query tree.

C. Algorithms for the General Case

In this subsection we consider briefly the case where the graph of query templates is not tree-based, but rather is a directed acyclic graph (DAG) or a graph with directed cycles. An example of the general case is the schema of Figure 1 with all query templates; an example of a DAG is obtained by removing `Q4` from that schema.

For non-tree-based query template graphs, the answers to one set of queries can provide the answer to another query. As a simple example, consider Figure 11, which shows a hybrid query tree corresponding to the query graph taken from Figure 1 omitting `Q4`. Given the intended semantics of

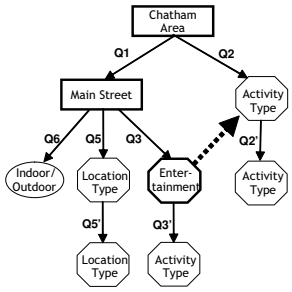


Fig. 11. Hybrid tree query assuming Q1, Q2, Q2', Q3 Q5, Q5', and Q6, and showing impact of one ground value on another node

the schema, if `Location` l is in `Region` r, and the user performs `ActivityType` a at l, then we may infer that the user performs a in r. That is, a pair of compatible answers to `Q1` and `Q3` can by used to derive an answer for `Q2`.

Consider now the general case of query graphs that are DAGs, and consider the problem of finding an optimal query sequencing for a user whose profile is known (analogous to the problem studied Subsection IV-A). Using a reduction to the Minimal Set Cover problem, it can be shown that in the case of DAGs, this problem is NP-complete. As a result, we cannot expect to find a polynomial time algorithm for the case of DAGs (or general graphs) analogous to Algorithm Instantiated-Tree-Based, nor analogous to Algorithm Generic-Tree-Based.

We have developed a heuristic algorithm for the generic case, which is a modification of Algorithm Generic-Tree-Based. One modification is to permit some flexibility about which succinct sequence is used in one step; rather than insisting on the maximally preferred sequence we allow for secondary factors to be used among the top p percent of preferred sequences. A second modification is to perform detailed book-keeping, so that the algorithm can keep track of query answers obtained directly from a user response, and also query answers that are derived from them.

V. Conclusions

This paper describes an end-to-end framework and prototype system, that combines statistical learning and explicit user queries to learn about user's locational and activity habits, in order to provide useful information for ad matching. This project, a part of the larger Bell Labs INA effort, is at an intermediate stage, and provides insight into two key areas. The first is the feasibility of using a "least common denominator" form of network observable information as the basis for statistical analysis for learning the regions that a user frequents. Specifically, we use the currently active cell ID (without signal strengths), which is available from essentially any cell phone. The second key area is the development of a generic approach for sequencing user queries, in order to fill out a user's profile information in a systematic way that maximizes the value to ad matching.

Key areas of future work in this project include further validation and refinement of the location clustering algorithms,

incorporation of Human Factors aspects into the query sequencing framework, and testing various components against user populations. It will also be interesting to apply the approach to query sequencing here to other domains where a mixture of statistical learning and explicit queries are appropriate, such as office assistance, personalization of services, and flexible workflow creation.

REFERENCES

[1] R. Dinoff, T. K. Ho, R. Hull, B. Kumar, D. Lieuwen, and P. Santos, "Intuitive network applications: Learning for personalized converged services involving social networks," *J. of Computers*, 2007.

[2] S. M. Virgin Mobile, "http://www.virginmobileusa.com/stuff/sugarmama.do."

[3] Blyk, "http://www.blyk.co.uk."

[4] J. Rekimoto, T. Miyaki, and T. Ishizawa, "LifeTag: WiFi-based continuous location logging for life pattern analysis," in *Third Intl. Symp. on Location and Context Awareness(LoCA 2007)*, Germany, Sep. 2007.

[5] A. LaMarca et. al. "Place Lab: Device positioning using radio beacons in the wild," in *Pervasive*, May 2005.

[6] A. Varshavsky et. al. "Are gsm phones the solution for localization?" in *IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2006)*, Semiahmoo Resort, WA, USA, Apr. 2006.

[7] E. Trevisani and A. Vitaletti, "Cell-id location technique, limits and benefits: An experimental study," in *IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2004)*, Lake District, UK, Apr. 2004.

[8] E. Meeuwissen, P. Reinold, and C. Liem, "Inferring and predicting context of mobile users," in *Bell Labs Technical Journal*, vol. 12, 2007, pp. 79–86.

[9] D. Ashbrook and T. Starner, "Using GPS to learn significant locations and predict movement across multiple users," in *Personal and Ubiquitous Computing*, vol. 7, London, UK, Oct. 2003.

[10] N. Marmasse and C. Schmandt, "Location-aware information delivery with commotion," in *Second International Symposium on Handheld and Ubiquitous Computing (HUC)*, vol. 1927, Bristol, UK, Sep. 2000.

[11] J. Hightower et. al. "Learning and recognizing the places we go," in *UbiComp 2005*, Tokyo, Japan, Sep. 2005.

[12] J. H. Kang, W. Welbourne, B. Stewart, and G. Borriello, "Extracting places from traces of locations," in 2^{nd} *ACM Intl. Workshop on Wireless Mobile Applications and Services on WLAN Hotspots (WMASH 2004)*, Philadelphia, PA, USA, Oct. 2004.

[13] K. Laasonen, M. Raento, and H. Toivonen, "Adaptive on-device location recognition," in *Pervasive*, 2004, pp. 287–304.

[14] M. Basseville and I. Nikiforov, *Detection of Abrupt Changes – Theory and Application*. Prentice-Hall, 1993.

[15] J. Sammon, "A non-linear mapping for data structure analysis," in *IEEE Transactions on Computers*, vol. C-18, 1969, pp. 401–409.

[16] A. Kobsa, "Generic user modeling systems," *User Modeling and User-Adapted Interaction*, vol. 11, no. 1–2, Mar. 2001.

[17] D. N. Chin, "Acquiring user models," *Artificial Intelligence Review*, vol. 7, no. 3–4, Aug. 1993.

[18] G. Stermsek, M. Strembeck and G. Neumann, "A user profile derivation approach based on log-file analysis," in *Intl. Conf. on Information and Knowledge Engineering (IKE)*, Las Vegas, USA, Jun. 2007.

[19] L. Aroyo, R. Denaux, V. Dimitrova, and M. Pye, "Interactive ontology-based user knowledge acquisition: A case study," in *European Semantic Web Conference (ESCW'06)*, Budva, Montenegro, Jun. 2006.

[20] J. Fink and A. Kobsa, "User modeling in personalized city tours," *Artificial Intelligence Review*, vol. 18, no. 1, Sep. 2002.

[21] R. Stegmann, "Improving explicit profile acquisition by means of adaptive natural language dialog," in *Doctoral Consortium, 10th Intl. Conf. on User Modelling*, Edinburgh, UK, Jul. 2005.

[22] R. Kass, "Building a user model implicitly from a cooperative advisory dialog," *User Modeling and User-Adapted Interaction*, vol. 1:1, Sep. 1991.

[23] M. Gervasio et. al. "Active preference learning for personalized calendar scheduling assistance," in *Proc. Intl. Conf. Intelligent User Interfaces*, 2005.

[24] N. Bila, J. Cao, R. Dinoff, T. Ho, R. Hull, B. Kumar, and P. Santos, "Algorithms for query sequencing in profile acquisition," Bell Labs Technical Manuscript, 2008, in preparation.

[25] N. Bila, J. Cao, R. Dinoff, T. Ho, B. Kumar, D. Lieuwen and P. Santos, "Intuitive Network Applications: Learning User Context and Behavior," Bell Labs Technical Journal, 2008, to appear.

[26] Google Maps with My Location, "http://www.google.com/gmm/mylocation.html."