Technical Report
Department of Distributed Systems, IPVS, University of Stuttgart

Privacy and Mobility: Optimized Training of Privacy-preserving Location-sharing Policies for Online Social Networks

Zohaib Riaz · Frank Dürr · Hasan Mohamed · Kurt Rothermel

Abstract To secure location privacy of social networks users in a manageable way, a large body of existing works focuses on automating the location-sharing decisions with online social contacts by defining machine-learning based *location sharing policies*. These policies are learned over actual user-responses to information sharing requests from their social connections under varying contextual conditions (e.g., time of day, location, etc.) and replicate user behavior autonomously for subsequent requests.

However, training high-performance location sharing policies, e.g., those with high accuracies, typically impose high computational demands. Hence training high-performance policies on mobile devices, which are computationally resource-constrained, is very challenging. In this regard, we propose a novel policy learning methodology whereby high-performance policies can be trained with low computational investment. This computational gain of our methodology is based on the selection of high quality training data. We also show that the high quality training data is composed of those training examples that are associated with strong mobility routine of the users. Thus we also propose an algorithm for efficient estimation of the dynamic strength of movement routine on mobile devices. Our evaluations on real world data demonstrate the usefulness of our approach.

Keywords Location privacy · Online social networks · Information sharing policies · Dynamic routine strength · Machine learning

Institute of Parallel and Distributed Systems, University of Stuttgart
Tel.: +49-711-685 88431
Fax: +49-711-685 88424
E-mail: {zohaib.riaz,frank.duerr,kurt.rothermel}@ipvs.uni-stuttgart.de, hasan.ibhasan@gmail.com

1 Introduction

With globally increased penetration of the internet and mobile phones, user-bases of online social networking platforms (OSNs), such as Facebook or Twitter, have seen tremendous rise. Sharing personal information with social connections on these platforms is, however, not without risk, in particular when it comes to personal privacy. It is well known that careless sharing of personal content in OSNs can lead to unintended revelations to others regarding ones current location, their health issues, personal views and inclinations, etc.

One way to minimize these privacy concerns is to avoid the publication of privacy-threatening personal content. Especially for scenarios where personal information such as location data may be shared frequently with others, a popular approach is to define *information sharing-policies* that mimic user sharing behavior [4, 10, 21, 24, 16, 17]. In the ideal case, a perfectly defined sharing policy relieves the user of performing manual sharing decisions while autonomously sharing their information with others in a *privacy-preserving* manner, based on the values of context variables such as time-of-day, location, etc. Specifically, no or little privacy-sensitive data is shared by such a sharing policy.

However, particularly for sharing location information, several studies show that the manual definition of such accurate privacy-preserving sharing policies is hard for OSN users [24, 17, 1]. Consequentially, a number of research works have tried to ease the process of defining sharing-policies, for example, by providing users with initial policy templates [21] or by employing machine-learning techniques for policy definition [24, 4] and its refinement [16, 10]. While these machine-learning (ML) based sharing policies show practical potential, their usability is severely limited by the high

computational demands of ML, especially on resource-constrained mobile devices.

A major reason for the high computational requirements of ML based policies is the need to perform *model selection*. Precisely speaking, many ML algorithms in their various possible parameter configurations, where each is termed a *model*, must be trained and validated to pick the one with the best estimated performance results for the learned policy. Naturally, this procedure can consume significant time and computational resources. However, a crucial property of the finally selected model is that it avoids *over-fitting*, i.e., it generalizes well to unseen data that may appear in actual policy usage after the end of the training phase.

To achieve good generalization of the learned models without the expensive model selection procedure, we take the option of improving training data quality. More specifically, we conjecture in this article that if *high-quality* training examples are selected for training, then computationally simple models that are prone to over-fitting can also yield policy performance comparable to that of the best model given by the model selection procedure. We define these high-quality training examples as the ones which, with high likelihood, represent the contextual situations for location sharing in actual policy usage. To concretely identify these examples, we use mobility analysis and define each example's quality as its strength of association with the daily user routine. In other words, training examples that are sampled along stronger routine movements, e.g., home to work, are deemed higher quality. The underlying intuition is simple: stronger daily routines typically do not change frequently and thus represent those contextual situations that will likely occur in actual usage of the trained sharing policies. Hence, learning users' sharing behavior regarding these important contextual situations by over-fitting them, using computationally simple machine-learning models, may still yield high policy performance.

To efficiently select the high quality examples on mobile devices, we also present a novel algorithm for real-time estimation of the dynamic strength of movement routine (DSR). We also evaluate the run-time performance of our DSR estimation algorithm as well as the privacy-utility performance of the correspondingly trained location sharing policies on a real-world dataset. Our results show that computationally simple policies can indeed perform well when training examples associated with strong user routine are used to learn them.

Overall, our contributions highlight the role of the DSR signal as a non-disruptive mobility-driven signal that can be practically sensed on resource-constrained mobile devices and can be exploited for efficient learning of effective location sharing policies on mobile devices.

We organize the remaining discussions in the article as follows. At first, we describe the related work in Section 2. In Section 3, we provide the problem statement followed by the detailed description of our DSR estimation algorithm in Section 4. In Section 5, we describe a computationally light-weight policy learning algorithm. Finally, we present the details of our implementations as well as the results in Section 6 before concluding in Section 7.

2 Related Work

In the following, we discuss two categories of existing works that relate to our contributions in this paper, namely, routine behavior detection and privacy-preserving sharing decisions.

2.1 Human routine behavior and predictability

Initial works attempted to extract *routine patterns* from mobility data [11,7]. At a course level, patterns in daily routine behavior, such as leaving for work in the morning, were identified by Katayoun and Gatica-Perez [11] using latent topic models (LDA) from collective mobility data of users in the Reality Mining dataset [6]. Eagle and Pentland [7] used principal component analysis for identifying the *eigen* routine movement patterns for individual users in the same dataset. Although, the routine patterns discovered by these approaches can be used to infer typical storlyines underlying user movements as well as for mobility prediction, they do not directly capture deviations of users from their routine behavior or the level of their predictability.

A more direct study in this regard was reported by Song et al. [25] in which they formulate the upper and lower bounds on the *overall* predictability of an individual's movements based on Shannon's entropy in their historical movements. Since this overall measure lacked detail, McInerney et al. [18] proposed a fine-grained measure to estimate the *momentary* value of entropy or unpredictability given the current user location, and so termed it as the *Instantaneous* entropy. This measure is based on Lempel-Ziv entropy estimators for time-series data and functions by computing the novelty in the recent sequence of visited locations with respect to the historical movement sequences. By ignoring temporal information associated with the movement sequences however, instantaneous entropy is unable to capture deviations in routine represented by temporal shifts in events, such as late arrival at work location, which may

affect information sharing behavior of users. Similarly, the effect of discontinuities in data (e.g., due to the unavailability of location information) on this measure is not explored. Nevertheless, the authors also propose a Dynamic Bayesian Networks (DBNs) approach, similar to [8], that additionally considers time and weekday information to estimate the strength of routine. The disadvantage of using DBNs, as stated by McInerney et al. [18], is the high computational cost of training user models (in their case using the Expectation Maximization algorithm) on resource-constrained mobile devices. We address this limitation by proposing a computationally efficient routine estimation algorithm that considers both, the spatial and the temporal aspects of user movements.

2.2 Privacy-preserving Information Sharing Decisions

In an effort to understand the reasoning behind location sharing decisions of users, various user studies have concluded that the person, social group, or the application that generates the location request represents the most important attribute affecting the sharing decision [19, 5]. The study by Toch et al. [26] suggests that users feel varying levels of comfort while sharing different types of locations. In particular, they found that privacy sensitive locations (e.g., home) are typically those that have low entropy in terms of the unique visitors and their corresponding visiting frequencies.

Significant research has also been invested into learning of sharing policies as a set of rules [24, 21, 2]. These rules allow users to express their willingness to share against intuitive contextual conditions defined over attributes such as the time-of-day. Sadeh et al. [24] and Bigwood et al. [3] additionally show that various machine-learning based models can be used to learn detailed sharing policies leading to higher policy accuracies. Fang and Lefevre [10] have proposed a privacy wizard that uses *active-learning* to incrementally learn users' sharing preferences by querying for their sharing behavior with the most "informative" friends. User behavior with these informative friends is then generalized to other social connections based on a detailed friendship graph, thus minimizing the user burden of manually elaborating privacy preferences with all of the social connections. A different approach that does not rely on the availability of a friendship graph was proposed by Bilogrevic et al. [4]. They use active-learning for refining sharing policies that are trained over a small number of training examples. Specifically, they ask the user to make those decisions where the model's prediction has high uncertainty. Moreover, they use *cost-sensitive* classifiers that can be tuned to be more sensitive to incorrect sharing privacy-sensitive information than incorrect suppression of non-privacy threatening information.

Our work fundamentally differs from the above machine-learning based approaches in the additional use of user-mobility based DSR signal. We contribute orthogonally to these approaches by helping simplify the otherwise compute-intensive model selection procedure which makes these approaches more viable on resource-constrained mobile devices.

3 Preliminary Definitions and Problem Statement

3.1 System Model

We consider the scenario of OSN usage where location information is shared by users with their social connections in either a pull- or a push-based fashion. In the pull-based version, users respond to location requests from their social connections in a reactive manner as opposed to the push-based version where they actively share their location.

We assume that the OSN users interact with the online platform using the OSN app-instances running on their location-enabled mobile devices. Whenever a user intends to share his location, voluntarily or as per request from a social connection, the OSN-app generates a location-request and forwards it to a trusted software agent (TSA), which also runs on the user's device. As similarly adopted in other works [4], we assume that the TSA has exclusive access to the devices' positioning technologies (e.g,. GPS) and acts as a gateway to location information for other on-device apps by servicing their location-requests.

3.2 Threat Model

We assume that privacy threats arise from unintended sharing of privacy-sensitive location events with social connections, which is known to cause various forms of social repercussions [20]. Moreover, we assume that the users may not fully trust the OSN platform-provider. Hence any given location-sharing decision made by the users includes privacy-threat considerations regarding both, the involved social connection and the OSN provider.

3.3 Protection Model

To protect against privacy threats from the social connections, the TSA implements a privacy protection

mechanism. For this article, we assume that this mechanism is a *location sharing policy* which represents location sharing preferences of the user towards his social connections (or any other location-based apps they use). The TSA initially learns the users' sharing preferences from their manual sharing choices and later enforces these choices autonomously by granting or denying incoming location requests.

To learn the sharing policy, the TSA acquires training samples over a limited period of time by asking users *in-situ questions*. Specifically, each question queries the user's willingness to share or deny their location information to various social contacts (typically as groups, e.g., friends or family) under the running situational *context*. Here, context is defined as a tuple including variables such as the time-of-day, day-type (weekday or weekend), as well as the geographic region. By aggregating user responses under different contexts, the TSA acquires the basic information for learning a sharing policy.

Recall that we assume that user-responses to the in-situ questions also take into account their privacy considerations regarding the OSN provider. Hence, learning sharing policies from these responses appropriately addresses the threat model described in Section 3.2.

3.4 Policy Learning Methodology

Technically seen, the process of learning the sharing policy from the limited training examples of a user presents a binary classification problem where sharing approvals should be discriminated from the denials based on the context variables. Inspired by visualizations from Hasti et al. [13], we depict this problem in Fig. 1. Here, several training examples representing the two classes, approvals as circles and denials as crosses, are shown in a 2D context space. In part (a) of the figure, we show the non-linear decision-boundary curve that, we assume, *perfectly* represents the *actual* user sharing preferences and dominantly separates the two classes. The few training examples that fall on the wrong side of this actual decision-boundary represent noisy training data.

In order to learn the actual decision-boundary from these training examples, choosing a suitable machine-learning (ML) algorithm is non-trivial. The best practices from the ML community suggest that different learning algorithms in various parameter configurations should be tried out, e.g., Decision-trees, Support Vector Machines (SVMs), etc., to choose the best among them for the given data (the model selection problem). If these steps are not followed properly, a fitted model may form a decision-boundary which is overly

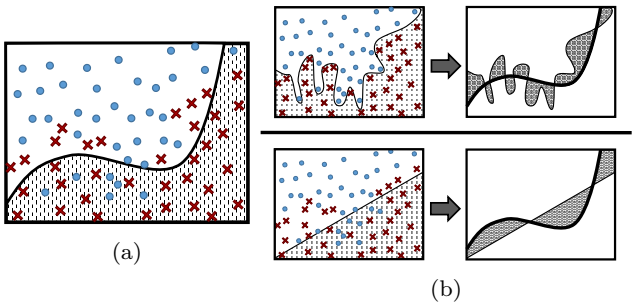


Fig. 1: (a) Actual Model. (b) Top row: Over-fitted model along with its misclassification space (shaded). (b) Bottom row: same for an under-fitted model.

flexible such that it fits even the noisiest training examples (over-fitting). At the other extreme, the fitted model may be too rigid (e.g., linear) to capture the actual (e.g., non-linear) decision-boundary. For the actual model shown in part (a) of Fig. 1, part (b) of the figure depicts the cases of over-fitting (top) and under-fitting (bottom). Based on the difference between the actual and the learned decision boundaries in these two cases, the figure also shows the corresponding shaded regions of the context space that can lead to misclassification (incorrect sharing decisions) as a result of over- and under-fitting.

To address the model selection problem, a popularly adopted methodology, for policy learning as well as other problems, is k-fold *cross-validation*. Here, the training data is split up into k equally sized parts or folds. For testing a particular ML algorithm with a particular parameter setting, i.e., a model, it is trained on the combined data of $k-1$ folds and then validated on the single remaining, *left-out* fold to simulate previously unseen data. This process is repeated k times for the same model where a different fold is left-out for validation in each run. By combining the validation results of the k runs, the performance of the model on the complete training data is determined. If m different models are tested in this manner to pick the best one, a total of at least $m*k$ train-validate cycles are required. Since a variety of ML algorithms exist and each possesses its own parameters, the number of different models m that need to be tried out, for approximating the best possible one, can grow large. For example, the Gaussian kernel-based SVMs used for policy learning by Bilogrevic et al. [4] have two parameters (C and γ), thus requiring a grid-search in a 2D parameter space.

3.5 Problem Statement

In this article, we study the potential value of a mobility-based signal, namely, the dynamic strength of movement routine (DSR), in enabling efficient learning of ML based location sharing policies on users' mobile devices.

To understand the main challenge of our work, consider the scenario where a user, say Alice, wishes to automate her location sharing decisions. To this end, she provides the initial set of training examples by answering various in-situ questions regarding her sharing preferences as posed by the TSA. At the end of this data collection phase, Alice triggers the TSA to initiate the training process and waits for its completion before activating the trained policy. For the TSA, the training process involves executing the expensive cross-validation (CV) procedure for model selection, as discussed above in Section 3.4, and then training the best model over Alice's complete training data. Given the limited computational resources of the mobile device, the training process could take a prohibitively long time, thus hurting Alice's interest and the usability of the automated sharing policy. A possible way of reducing the policy training time is to offload the expensive CV operation to a cloud based service. Doing so however requires Alice to trust the service provider with her contextual data as well as her detailed location sharing preferences, which may be undesirable.

As part of the solution to this problem, we conjecture that the computationally expensive CV procedure may not be required if we can improve training data quality. We explain this in reference to the top row of Fig. 1(b). Here, the flexible model chaises the noisy training data to form a fluctuating decision boundary. Although this model perfectly fits the training data, it may perform poorly in actual usage due to the significant size of the misclassification space. If however the noisy training examples are removed, the magnitude of the fluctuation of this model's decision boundary will decrease. This in turn will conform the model more to the actual one shown in Fig. 1(a), thus reducing the size of the misclassification space. Simply stated, we make the following claim:

If noisy training examples, that are less likely to repeat in actual usage and thus are mis-representative of the distribution of training data in the context space, are excluded from the training data, then, an over-fitted model may still perform comparably to a model that is selected by using the CV procedure.[1]

The implication of this claim is that the computational load of training an accurate sharing policy by the TSA can be brought down to *training a single model using less noisy training data* in contrast to the time-consuming training of a plethora of models using CV.

More formally, let the set $E = \{e_1, ..., e_q\}$ represent the set of all training examples collected by the TSA. Let $\mathcal{N} = \{\eta_1, ..., \eta_q\}$ be the "noise-confidence set" where $\eta_i \in \mathcal{N}$ represents the confidence with which the training example $e_i \in E$ can be deemed noisy, with $\eta_i = 1$ implying full confidence. Then, a subset of high quality training examples $E_\theta \subseteq E$ may be obtained such that for each $e_i \in E_\theta$, $\eta_i < \theta$. Here, θ represents the confidence associated with the most noisy sample that is included in E_θ. Using E_θ, a location sharing policy $\mathcal{M}_\theta$ can be learned by employing an over-fitting-prone algorithm, in compliance with our above claim. In contrast, a CV based model $\mathcal{M}_{CV}$ may be trained over the full dataset E. Let the performance difference between the two models be Δ_{test}.

$$\Delta_{test} = \phi_{test}(\mathcal{M}_\theta) - \phi_{test}(\mathcal{M}_{CV}) \tag{1}$$

where $\phi_{test}(\mathcal{M})$ represents the performance of model $\mathcal{M}$ in actual usage.

We hypothesize that if θ is decreased to a *safe-value*, where most noisy training examples are left out of E_θ, then Δ_{test} will approach zero. Note that it is important to precisely determine this safe-value as decreasing θ further will leave out too many training examples from E_θ, which may also adversely affect the performance of $\mathcal{M}_\theta$ by lowering its coverage of the context space. Thus we require that the noise-confidence values $\eta_i \in \mathcal{N}$ should vary over a well-defined interval with a meaningful interpretation. Moreover, $\eta_i \in \mathcal{N}$ should be efficiently computable on today's mobile devices.

To meet these challenging requirements, we use the mobility-based DSR signal as a proxy to approximate $\eta_i \in \mathcal{N}$ associated with $e_i \in E$. As also motivated in Section 1, we deem those training examples as more noisy that are associated with rare user behavior, i.e., a high deviation from users' mobility routines, and are less likely to repeat in actual usage. In contrast, training examples that are sampled when the users are in their stronger mobility routines are deemed less noisy and more representative of the actual contextual situations. Since the existing DSR estimation algorithms are not suitable for mobile devices (cf. Section 2), we first present our novel, efficient DSR estimation algorithm in the next section.

[1] Note that this claim is not valid for under-fitted models as, in their case, the size of their misclassification space is mainly contributed by their inflexibility compared to the actual model.

4 The Routine-strength Estimation Algorithm

In this section, we describe our metric for the mobility based DSR signal and present a detailed algorithm for its computation. Initially however, we briefly discuss the concept of conditional trajectory entropy as introduced in previous works [9,14] and explain why it is *not* applicable as a direct metric for DSR. Note that for the rest of the paper, we use the terms *trajectory* or *path* interchangeably to refer to the same concept.

4.1 Conditional Trajectory Entropy

For a given user, we assume that there exists a finite set V of geographical locations, e.g., home region, work region, etc., where they spend time as part of their daily movements. We further assume that user mobility between these states can be modeled as a Markov chain whereby the the probabilities of transition between any two states $P_{x_t x_{t+1}}$ is given by a single transition matrix A. Using A, the probability of a particular movement trajectory $x_{sd} = \{x_1 = s, ..., x_k = d\}$ between a start location s and destination location d is given by:

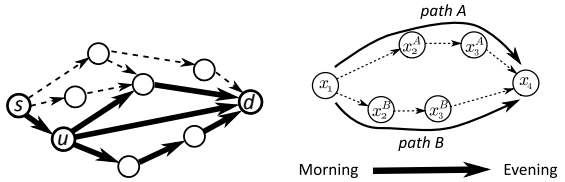
$$p(x_{sd}) = \prod_{t=1}^{t=k-1} P_{x_t x_{t+1}} \tag{2}$$

In [14], Kafsi et al. also consider the case of an aperiodic and irreducible finite-state Markov chain. They further define the set $\mathcal{X}_{sd}$ of trajectories (of all possible lengths) from s to d where the sum of the probabilities of all $x_{sd} \in \mathcal{X}_{sd}$ is 1. The entropy of the random variable representing the possible user trajectories X_{sd} between s and d is then defined as follows.

$$H_{sd} = H(X_{sd}) = -\sum_{x_{sd} \in \mathcal{X}_{sd}} p(x_{sd}) log(p(x_{sd})) \tag{3}$$

Intuitively seen, H_{sd} represents the uncertainty associated with the actual trajectory taken by the user. Kafsi et al. [14] consider the advanced case where it is additionally known that the user traversed a set of intermediate states $\boldsymbol{u}$ between s and d. Thus they define the *conditional* trajectory entropy $H_{sd|\boldsymbol{u}}$ to determine the remaining uncertainty in user trajectory by only considering those trajectories that pass through the states $\boldsymbol{u}$, i.e., $\mathcal{X}_{sd|\boldsymbol{u}}$. In Fig. 2 (a), we provide an example illustration of the path links that lead from s to d. Here, the links that form the conditional trajectories passing through the node u are shown in bold (solid) lines.

In an extended work, Kafsi et al. [15] show that by comparing the trajectory entropy H_{sd} with the *conditional* trajectory entropy $H_{sd|\boldsymbol{u}}$, one can determine the affect of revealing the intermediate locations $\boldsymbol{u}$ on the predictability of the remaining user trajectory. To perform this comparison, they determine the *entropy ratio* $\alpha = H_{sd|\boldsymbol{u}}/H_{sd}$. If $\alpha < 1$, it indicates a decrease in uncertainty (increased predictability) regarding the actual trajectory taken by the user as a result of publishing $\boldsymbol{u}$, whereas $\alpha > 1$ indicates the drop in trajectory predictability.



(a) (b)

Fig. 2: (a) All Markov paths between states s and d. Links that form part of the paths conditioned on state u are shown in bold. (b) Example of two paths between morning and evening locations of a user.

4.2 A novel DSR Metric

It is not difficult to imagine that the rise and drop in the predictability of user movements can be associated to their routine and out-of-routine behavior. Hence, the entropy ratio α seems a feasible metric for determining DSR. However, using α for this purpose presents several drawbacks. For instance, computing $H_{sd|\boldsymbol{u}}$ based on the algorithm proposed by Kafsi et al. [14] is computationally expensive (complexity $\mathcal{O}(|\boldsymbol{u}|N^3)$ where N is the number of states of the Markov chain). This is because they allow cyclic paths between s and d, leading to (possibly) an infinite total number of paths $|\mathcal{X}_{sd}|$ or $|\mathcal{X}_{sd|\boldsymbol{u}}|$ between s and d. Moreover, use of α as a DSR metric presents further drawbacks which we highlight with the help of the following examples. We also use these examples to motivate our novel DSR metric called the *relative path strength*.

Example 1: Consider a user Bob who, among other paths, follows a strong *home→ work→home→other* routine during the day. If we assume his start location s as his home in the morning and the destination location d as his home location in the evening, then being at work location during the middle of the day represents being on a popular path between s and d, thus indicating in-routine behavior. Conversely, if Bob is found at another location during the day, out-of-routine behavior should be indicated.

The above example highlights the drawbacks of some of the assumptions made by Kafsi et al. [14] in defining the α metric. At first, human movement preferences may not be the same for different times of the

day (e.g., transitions to work-place are more probable in the morning). Secondly, the movement trajectory of Bob over a day consists of a relatively small number of hops between different states unlike the possibly large number of hops introduced by infinite cyclic paths as considered in Kafsi's approach. In general, instead of considering only spatial information, we also wish to consider the temporal dimension in our calculation of DSR, which makes Kafsi's algorithm unsuitable in our case.

Example 2: As illustrated in Fig. 2 (b), consider another user Alice who traverses two main paths, A and B, between her morning and evening locations ($s = x_1$ and $d = x_4$ respectively). Let path A represent Alice's strong daily work routine and path B her relatively weak monthly routine of day-time short business visits. Assuming that both these paths are strictly followed if taken (no branching to other possible states), then computing α anywhere along the two paths (states x_2^A, x_3^A, x_2^B, or x_3^B) results in the same 0 value. This occurs because conditioning on any of these intermediate states only leaves a single path to d (perfect predictability). However, the same high predictability value for being on paths A and B contradicts Alice's different strength of routine associated with path A (high strength) and path B (low strength).

For our purpose of differentiating movements based on their strength, we require a metric that also takes into account the likelihood of the traversed path. More precisely, we require a comparison of the probabilities of the conditional paths $\mathcal{X}_{sd|\boldsymbol{u}}$ that pass through $\boldsymbol{u}$ with those of the remaining paths $\mathcal{X}_{sd|\bar{\boldsymbol{u}}}$.

To this end, we take into account both, the spatial and the temporal information associated with user movements, by defining a mobility model based on discrete *time non-homogeneous* Markov chains. Hence, we divide the time of day into D equal-length slots, e.g., hourly, and compute a distinct transition matrix A_t for each time-slot t from a user's location information over a learning period. Doing so implies that the paths $\mathcal{X}_{sd}$ leading from a particular start location $s_{t_s} \in V$ at time-slot t_s to particular destination $d_{t_d} \in V$ at time-slot t_d are all of equal length, i.e., $t_d - t_s$. Also, the sum of the probabilities of paths leading from all possible start locations in t_s to all possible destination locations in t_d (where each is computed using Eq. 2) sum to 1. Notably, for a given single start and destination pair, the sum of the probabilities of all paths between them may not sum to 1.

We now define our novel DSR metric below while ignoring time subscripts, i.e., t_s or t_u, for readability reasons.

Definition 1 Relative Path Strength (RPS): Let $\mathcal{P}_{sd|\boldsymbol{u}}$ denote the set of probabilities of all conditional paths $\mathcal{X}_{sd|\boldsymbol{u}}$ that pass through $\boldsymbol{u}$, and let $\mathcal{P}_{sd|\bar{\boldsymbol{u}}}$ denote the set of probabilities for all remaining non-conditional paths $\mathcal{X}_{sd|\bar{\boldsymbol{u}}}$ between s and d. Moreover, let $\hat{\mathcal{P}}_{sd|\boldsymbol{u}}$ and $\hat{\mathcal{P}}_{sd|\bar{\boldsymbol{u}}}$ denote the corresponding normalized probabilities for the sets $\mathcal{P}_{sd|\boldsymbol{u}}$ and $\mathcal{P}_{sd|\bar{\boldsymbol{u}}}$, respectively, which sum to 1. Then Relative Path Strength is defined as follows:

$$RPS(\mathcal{P}_{sd|\bar{\boldsymbol{u}}}, \mathcal{P}_{sd|\boldsymbol{u}}) = \sum_{i=1}^{|\mathcal{P}_{sd|\bar{\boldsymbol{u}}}|} \sum_{j=1}^{|\mathcal{P}_{sd|\boldsymbol{u}}|} \hat{p}_i \cdot \hat{p}_j \cdot (p_i - p_j) \quad (4)$$

where $p_i \in \mathcal{P}_{sd|\bar{\boldsymbol{u}}}$, $\hat{p}_i \in \hat{\mathcal{P}}_{sd|\bar{\boldsymbol{u}}}$, $p_j \in \mathcal{P}_{sd|\boldsymbol{u}}$, and $\hat{p}_j \in \hat{\mathcal{P}}_{sd|\boldsymbol{u}}$.

Hence RPS computes the expected difference in probabilities of the paths belonging to the sets $\mathcal{X}_{sd|\bar{\boldsymbol{u}}}$ and $\mathcal{X}_{sd|\boldsymbol{u}}$. The RPS metric varies in the range $-1 \leq RPS \leq 1$ considering the two extremes of very high probability conditional paths ($RPS = -1$ or high routine behavior) or very low probability conditional paths ($RPS = 1$ or out-of-routine behavior) . With reference to Example 2 and Fig. 2 (b), let path A and path B have probabilities $p_A = 0.9$ and $p_B = 0.1$ respectively. Then conditioning on any state on path A yields $RPS = -0.8$ whereas the same on path B gives $RPS = 0.8$ which conforms with Alice's movement behavior on these paths. For ease of interpretation, we define DSR as a scaled version of RPS as follows:

$$DSR = (RPS + 1)/2 \quad (5)$$

Hence $0 \leq DSR \leq 1$ where strong routine is indicated by $DSR = 0$ and vice versa. Note that as DSR approaches a value of 0.5, it indicates equally likely conditional and non-conditional paths and thus the lack of evidence to call the current user movement as out-of-routine. Note also that since the number of paths in $\mathcal{X}_{sd|\bar{\boldsymbol{u}}}$ and $\mathcal{X}_{sd|\boldsymbol{u}}$ can vary, the conditional path probabilities by themselves, without looking at them in a relative manner as RPS does, cannot be directly used to compute DSR.

4.3 DSR Estimation Algorithm

To estimate DSR for any given location u_t of a user, we now present our detailed methodology regarding the following: (1) selection of appropriate start and destination locations and (2) efficient computation of the probability sets $\mathcal{P}_{sd|\boldsymbol{u}}$ and $\mathcal{P}_{sd|\bar{\boldsymbol{u}}}$. Based on these probabilities, we can compute DSR according to Eq. 5.

4.3.1 Selecting start and destination locations

To understand the significance of choosing the right start and destination locations, consider again the scenario sketched in Example 2 and Fig. 2 (b). Assume that we want to compute DSR at path node $u_t = x_3^B$. To compute $DSR(u_t = x_3^B)$, we can set destination d as x_4 and could choose between x_2^B or x_1 as the start location s. If $s = x_2^B$, then path $\{x_2^B, x_3^B, x_4\}$ represents the strongest path between the chosen (s, d) pair, conveying the wrong impression that Alice is in her strong routine (although she is on path B, i.e., her monthly business trip). Considering $s = x_1$ avoids this local view of Alice's mobility by bringing path A into consideration, thus, yielding the correct estimation of routine strength due to her being on path B.

The principle we derive out of the above example is that the start and destination should represent the major sources and sinks, respectively, of the user's daily movements. By anchoring the computation of DSR on these sources and sinks, we intend to compare the user's current movement between his regularly visited locations. To find such locations, we collect per-time-slot probabilities π_t regarding the occurrences of different locations from the set V for each of the D time-slots of the day from the user's data. Using π_t, we compute the entropy $H(x_t)$ of the random variable representing user location x_t in time-slot t (defined similarly as Eq. 3). A low value of entropy indicates a high non-uniformity of the probability distribution over possible locations in a time-slot t, thus indicating source/sink locations. Given that we wish to compute DSR at time t_u, we search within $|k_{max}|$ time-slots on both sides of t_u for the ones that minimizes $H(x_t)$, in order to get the start location's time-slot t_s and the destination location's time-slot t_d. Formally, we determine t_s and t_d as follows.

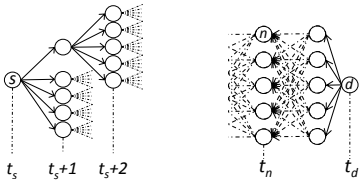
$$t_s = t_u + \underset{\forall k \in [-k_{max}, -1]}{\operatorname{argmin}} \; H(x_{t+k}) = f(\pi_{t+k}) \tag{6}$$

$$t_d = t_u + \underset{\forall k \in [1, k_{max}]}{\operatorname{argmin}} \; H(x_{t+k}) = f(\pi_{t+k}) \tag{7}$$

Having determined t_s and t_d, the set of possible start and destination locations, V_{t_s} and V_{t_d}, along with their probabilities are given by π_{t_s} and π_{t_d} respectively. Knowing t_s and t_d also allows us to determine the set of intermediate states $\boldsymbol{u} = \{u_{t_s+1}, ..., u_t\}$ that are already traversed by the user.

4.3.2 Optimal Population of Path-probability sets $\mathcal{P}_{sd|\boldsymbol{u}}$ and $\mathcal{P}_{sd|\bar{\boldsymbol{u}}}$

To simplify the following discussion, we select the single most probable start location $s \in V_{t_s}$ and similarly a single destination location $d \in V_{t_d}$. For the same reason, we also ignore the cyclic nature of time-slots, such as their overflow to 0 as a result of increments, in our use of the formal notation.



(a) (b)

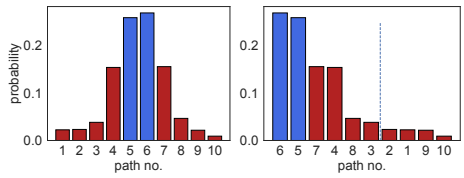
Fig. 3: (a) Exponentially expanding path tree. (b) Trellis diagram of Markov paths growing backwards from d.

Using the transition matrices A_t, we can compute the total probability of moving from any state x_{t_s} at time-slot t_s to any state x_{t_d} at time-slot t_d through all possible paths as follows.

$$P(x_{t_d}|x_{t_s}) = a_{x_{t_s}, x_{t_d}} \in A_{t_s t_d} \text{ where } A_{t_s t_d} = \prod_{t=t_s}^{t_d - 1} A_t \tag{8}$$

However, computing individual probabilities of all of the paths that make up $P(x_{t_d}|x_{t_s})$ is non-trivial. As shown in Fig. 3 (a), these paths grow exponentially as a tree with each hop from s towards d. Precisely speaking, the total number of paths between s and d are $N^{\delta t}$ where $N = |V|$ is the number of Markov chain states and $\delta t = t_d - t_s$ is the length of each path between s and d. Based on the well-established regularity of human movements [25], one could argue that the tree of possible paths may not grow exponentially due to sparsness of the transition matrices A_t. However, practical considerations in computation of A_t, such as deliberate infrequent sampling of location for energy-efficiency on mobile devices or unavailability of position fixes, forces the adoption of parameter smoothing techniques (e.g., additive smoothing). Hence, typically all entries of matrices A_t are non-zero, leading to exponential growth (with δt) of the number of paths and, therefore, the size of the probability sets $\mathcal{P}_{sd|\boldsymbol{u}}$ and $\mathcal{P}_{sd|\bar{\boldsymbol{u}}}$ as well as their corresponding normalized probability sets $\hat{\mathcal{P}}_{sd|\boldsymbol{u}}$ and $\hat{\mathcal{P}}_{sd|\bar{\boldsymbol{u}}}$.

To efficiently compute these probability sets, we take the following steps. We note that if the expansion of possible paths is limited to those that make up the major probability mass of $\hat{\mathcal{P}}_{sd|\boldsymbol{u}}$ and $\hat{\mathcal{P}}_{sd|\bar{\boldsymbol{u}}}$, we can still approximate RPS, and hence DSR, with good accuracy. This is because RPS computes the expected



(a) Set $\mathcal{P}_{sd}$ (b) Sorted set $\mathcal{P}_{sd}$

Fig. 4: Probabilities in the set $\mathcal{P}_{sd} = \mathcal{P}_{sd|\boldsymbol{u}} \cup \mathcal{P}_{sd|\bar{\boldsymbol{u}}}$. The blue and red colors denote the conditional and the non-conditional paths respectively.

difference between the two probability sets where contribution of small probabilities in both sets is quadratically minimized due to their multiplication (see Eq. 4).

To elaborate our solution further with an example, Fig. 4 (a) shows the probability distribution over the conditional (blue) and non-conditional (red) paths between a given s and d. Part (b) of Fig. 4 also shows the sorted version of this distribution. Accordingly, to optimally populate the probability sets $\mathcal{P}_{sd|\boldsymbol{u}}$ and $\mathcal{P}_{sd|\bar{\boldsymbol{u}}}$, we explore these paths in descending order of their probabilities. We stop path exploration when the difference in probabilities of the latest explored path and its predecessor is consistently low, i.e., for a threshold number of explored paths which define the stopping condition. Thus we do not consider these equal (and typically low) probability paths, for instance paths $\{2, 1, 9, 10\}$ onwards in Fig. 4 (b), which may only exist due to parameter smoothing, without significantly harming our estimate of RPS and, thus, DSR.

4.3.3 Path Exploration Algorithm

For optimal expansion of the path-tree, we perform A* strategy based node expansion beginning at the start location at time t_s, i.e, s_{t_s}. On expansion of a path-node at time-slot t_{n-1}, each of its child node n_{t_n} is put into a priority queue along with its *weight* as given by a function $f(n_{t_n})$. For a subsequent expansion, the node with the highest weight among those present in the priority queue is chosen.

To ensure that the highest probability paths are explored first, the weight function, $f(n_{t_n}) = g(s_{t_s}, n_{t_n}) + h(n_{t_n}, d_{t_d})$, sums the two-way path probabilities around n_{t_n}. Precisely speaking, $g(s_{t_s}, n_{t_n})$ represents the associated path's probability of traversing from node s_{t_s} to node n_{t_n} which is trivially computed using Eq. 2. In contrast, the heuristic $h(n_{t_n}, d_{t_d})$ represents the highest path probability of reaching the destination node d_{t_d} from n_{t_n}. *If we guarantee that $h(n_{t_n}, d_{t_d})$ is never under-estimated (or is admissible)*, then node expansion prioritized by $f(n_{t_n})$ effectively leads to the exploration of the highest probability path to d_{t_d} first. In other words, if for the weights of two nodes a_1 and a_2, $f(a_1) > f(a_2)$, then it is not possible for a child node of node a_2, say a_3, that $f(a_1) < f(a_3)$. Hence prioritizing a_1's expansion between the two nodes is optimal.

To compute $h(n_{t_n}, d_{t_d})$ while avoiding its under-estimation, we perform a backward-expansion beginning from the destination d_{t_d} (see Fig. 3 (b)). More specifically, for each backward hop from time-slot $t+1$ to t, we record for each node n_t at time-slot t the probability of only the best path reaching it from time-slot $t+1$ based on the Viterbi Algorithm [12]. Thus effectively, we store for each node n_t the value of $h(n_{t_n}, d_{t_d})$, i.e., the highest probability path leading to d_{t_d}. For k backward hops where $k <= D$ (a day has D time-slots), the complexity of computing $h(n_{t_d-k}, d_{t_d})$ is $\mathcal{O}(kN^2)$. Since this computation is not specific to a particular start location s or the intermediate path states $\boldsymbol{u}$, it can be pre-computed once for all possible destinations for each of the D time-slots of a day. In our evaluations (cf. Section 6), we will also report the time taken by this pre-computation as well as by our optimized A*-based path exploration on typical mobile devices.

For a single (s, d) pair, we run the above path-exploration algorithm twice, once for each probability set $\mathcal{P}_{sd|\bar{\boldsymbol{u}}}$ and $\mathcal{P}_{sd|\boldsymbol{u}}$. For $\mathcal{P}_{sd|\bar{\boldsymbol{u}}}$, we populate the set $\mathcal{P}_{sd}$ and then remove any paths passing through all of the intermediate states $u_{t_u} \in \boldsymbol{u}$. Note that we make a separate second run of the path-exploration algorithm for $\mathcal{P}_{sd|\boldsymbol{u}}$ as it is possible that the conditional paths have relatively lower probabilities and are not sufficiently explored during the population of set $\mathcal{P}_{sd}$. Thus, in the second run, we enforce the tree expansion only to the node $u_{t_u} \in \boldsymbol{u}$ for expansions from time-slot $t_u - 1$ to t_u. Such enforcement in turn requires careful use of the pre-computed heuristic $h(n_{t_n}, d_{t_d})$ for correct computation of the weight function $f(n_{t_n})$. In general, we guide the path expansion for a node n_{t_n} in the path-tree till the *next known node*. For instance, if $\boldsymbol{u} = \{u_{t_u}\}$, then the next node can be u_{t_u} if $t_u > t_n$, or d_{t_d} if $t_n \geq t_u$. Hence, for these two cases, we compute the correct value of the heuristic as either $\big(h(n_{t_n}, u_{t_u}) + h(u_{t_u}, d_{t_d})\big)$ or $h(n_t, d_{t_d})$ respectively.

4.3.4 Expected DSR value

Until now, we have explained the computation of DSR for a single pair of start and destination locations. As discussed in Section 4.3.1, we determine not one but a set of possible start and destination locations, $d \in V_{t_s}$ and $d \in V_{t_d}$, respectively. Thus the overall $DSR(u_t)$ for the current location u_t is an expectation as defined

below.

$$DSR(u_t) = \sum_{s \in V_{t_s}} P(s) \sum_{d \in V_{t_d}} P(d_{t_d}|s_{t_s}) DSR(u_t|(s, d, \boldsymbol{u})) \tag{9}$$

Here $P(s)$ represents the probability of the start location as per time-slot probabilities π_{t_s}, and $P(d_{t_d}|s_{t_s})$ is computed using Eq. 8. Both of these probabilities are also part of the pre-computations that need to be performed only once. Since the source and sink locations that form the start and destination locations may vary on weekday and weekends, we determine two versions of π_t, i.e., π_t^{wd} and π_t^{we}. Hence, our algorithm can capture the variation of user-routines between weekdays and weekends by appropriately using π_t^{wd} or π_t^{we} in determination of $P(s)$.

5 A Light-weight Policy-learning Algorithm

As discussed in detail in Section 3.5, we want to validate our hypothesis that high-performance location sharing policies can be trained using a simple over-fitting-prone learning algorithm over a subset of less noisy training examples $E_\theta \subseteq E$. Each example $e_i \in E_\theta$ is a tuple (*predictors*, *target class*) representing a location request from a social connection. In this article, we limit the *predictors* to the tuple (t, w, l), where t is the hour-of-day, w is the day-type (weekday or weekend), l is the location-ID. Correspondingly, the *target class* represents the user-response, i.e., positive (request granted) or negative (denied). We now briefly describe a computationally simple approach to learn a basic policy called the Location-sharing Rules (LR) which is prone to over-fitting the training data.

The *training process of LR*, as discussed by Benisch et al. [2], involves a straight-forward aggregation of the positive (q) and the negative responses (r) of the user against each unique combination of context-predictors (t, w, l). These aggregates can be maintained inside a look-up table LT_{main} for each social group.

To make a sharing decision using LT_{main} for a location request from a certain social group, the associated context tuple is used to look-up the aggregates, i.e., $(|q|, |r|) = LT_{main}(t, w, l)$, and the request is granted if $score(main) > 0$. Here, $score(main)$ is defined as $(|q| - c * |r|)$, where c represents the user-defined relative cost of revealing a private event in contrast to the suppression of a non-private event.

Note that each row of the look-up table in conjunction with the score function is seen as a rule. Moreover, presence of even a single positive event for a given context-tuple in the training data, i.e., $|q| = 1$ and $|r| = 0$, leads to the formation of an additional rule, thus making the LR policy over-fitting prone.

Obviously, the limited number of training examples in the set E_θ may not contain user-responses regarding each (t, w, l) combination. Thus the LR policy is unable to answer those *unseen* requests which were not part of the training data. To overcome this limitation, we additionally build *marginalized lookup-tables*, LT_t, LT_w, and LT_l, of aggregates for the individual context-predictors t, w, and l, respectively, during the training phase. Then a request which is unseen in LT_{main} is granted if the following condition is satisfied.

$$min\big(score(t), score(w), score(l)\big) > 0 \tag{10}$$

If the request is also not found in one of these three marginalized tables, then the score of that table is left out of the above equation. Hence if a request is also marginally unseen, it is denied.

6 Implementation and Evaluations

In this section, we first describe the dataset for our evaluations as well as its necessary pre-processing steps. Then we present two kinds of results regarding: (1) The run-time performance of the DSR estimation algorithm on mobile devices and its intuitive interpretability. (2) Performance comparison between the LR policy (cf. Section 5), that is trained using DSR-based selection of training examples, and existing machine-learning based policy-learning approaches.

6.1 The Dataset and its Pre-processing

For our evaluations, we needed a long duration real-world dataset that not only captures fine-grained location information for a number of subjects but also the information regarding their location-sharing preferences towards their social connections. In particular, we require the concrete events that represent the in-situ ground-truth about their sharing behavior (share/deny) for several location requests. While several location datasets are openly available, unfortunately none of them contains the detailed location sharing behavior of subjects to the best of our knowledge. To overcome this shortcoming, we instead opt for the 9 month long Reality Mining dataset from MIT [6], which, apart from location information, also provides the call logs of its 94 subjects. Thus we use call-log information as a proxy for the subjects' sharing behaviors, which is well-substantiated by existing research.

For instance, several user-studies affirm that the *frequency of communication* between people (especially phone call based) positively correlates with their mutual *emotional closeness* [27,23,28]. In turn *emotional closeness* has been found to be the best indicator of *willingness to share personal information*, including location data, in online social networks [28]. Based on these results, we process the call-logs of the dataset subjects to determine their willingness to share information with their contacts, in order to represent their ground-truth sharing behavior. Apart from the frequency of communication, we also take into account other factors to determine users' sharing behaviors such as the time of calls and their direction (incoming/outgoing), as discussed in [28], as well as call-duration.

Before describing the pre-processing of call-logs in more detail, we emphasize that the inferred location-sharing ground-truth behavior does not affect our results as we provide a comparison between different algorithms on this same data to demonstrate our contribution.

6.1.1 Inferring Sharing-behavior from Call-logs

For each call in the subjects' call-logs, the dataset provides its time, duration, direction (missed/incoming/outgoing), and the hash of the other person's number. We classify each call event as either a *positive* (share) or a *negative* (deny) sharing event with the other calling person, or their social connection (SC). To this end, we first infer the *relative willingness*, denoted $\omega(sc_i|\boldsymbol{SC})$ with value range $[0,1]$, of the subject to share his location with a particular connection sc_i out of all his social connections $\boldsymbol{SC}$ as being proportional to the following features: (1) mutual calling frequency, (2) proportion of outgoing to incoming calls with sc_i, (3) mean call-duration with sc_i, (4) total number of mutual calls.

Furthermore, we also decide a *per-call positivity score*, denoted $\zeta(c_j|\boldsymbol{C_{sc_i}})$ with value range $[0,1]$, for each call $c_j \in \boldsymbol{C_{sc_i}}$ associated with the social connection sc_i based on proportionality with: (1) the relative duration, and (2) the hour-of-day based probability, of call c_i compared to the other calls in $\boldsymbol{C_{sc_i}}$.

The overall positivity score associated with a call c_j from sc_i is then given by the product $\rho(c_j) = \zeta(c_j|\boldsymbol{C_{sc_i}}) \cdot \omega(sc_i|\boldsymbol{SC})$. Finally, we set the top γ percent of each subject's calls, according to the positivity score $\rho(c_j)$, as his positive sharing events. Here γ denotes the openness of the subject towards others and is estimated as the proportion of his overall outgoing calls irrespective of the SCs. The remaining calls are set as negative events. On average, we identified 70 % calls as positive sharing events.

6.1.2 Inferring Social Groups

For each subject, we identify a limited set, typically $2-3$, of their social groups to simplify the complexity of policy definition. We achieve this by first clustering each subject's calls into k groups (we try different values of k for each subject). Specifically, we use k-means clustering to group the calls based on their temporal features (time-of-day and weekday-or-weekend) and the subject's willingness to share information with the other participant $\omega(sc_i|\boldsymbol{SC})$, as determined above. Then, each social connection sc_i is assigned to that particular call group-ID, or social group-ID, to which most of his calls belong.

6.1.3 Location-data Pre-processing

Location information is represented by the frequent time-stamped transitions between cell-towers, each with unique ID, that are sensed by the mobile devices. Using this data, we identify various high level locations visited by the subjects. We also filter out subjects who, on average for a single day, have less than 3 location visits or less than a single call event, or they have a data recording duration of less than 1 month. For the remaining 68 subjects, we map their location visits to the underlying hourly time-slots of the day ($D = 24$). Visualizing plots of time-slotted days, as shown in Fig. 5, hinted one or more re-locations for a number of subjects.

For a fair evaluation of sharing policy performance, we keep the largest contiguous portion of each subject's data that belonged to a single region involving no relocation. This operation leaves us with, on average for a single subject, 21 unique locations and 651 calls (22 hours of calling time) over 66 days of data. Figure 5 also highlights the fact that our dataset exhibited significant amounts of missing location data, thus necessitating the use of parameter smoothing techniques, e.g., for estimating transition matrices A_t. For computing A_t from the time-slotted location data, we use maximum likelihood estimation as described in detail in [22].

6.1.4 Training and Test Dataset

In both types of evaluations that follow, namely, regarding DSR estimation and sharing policy learning, we use the initial 40 % of the total call data for each subject, and its associated time-slotted location information, as the training set. The remaining data is set apart for

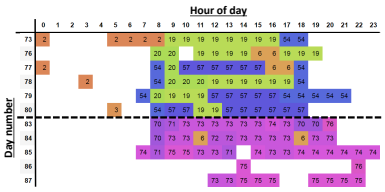


Fig. 5: Time-slotted days of a subject. Each time-slot shows a uniquely colored location-ID. A sharp overall change of colors after day 80 marks a relocation.

testing the trained algorithms. For DSR computation, we fix the vector of intermediate states to the current location, i.e., $\boldsymbol{u} = \{u_t\}$. Doing so allows us to judge the DSR value at u_t independently of the DSR values at previous trajectory locations. Note that this setting does not represent an advantage for our algorithm in terms of performance as it actually has to explore more paths (is less constrained) in computing $\mathcal{P}_{sd|\boldsymbol{u}}$.

6.2 Results for DSR

We first present the performance and the quality-related results for the estimated DSR signal.

6.2.1 Performance on Mobile Devices

To test the performance of our algorithm, we use a Samsung Galaxy S5 smartphone (Quadcore 2.5 GHz CPU, 2GB RAM). We divide the overall computation of the DSR signal into a *pre-computation* and an *online* phase. In the former, a one-time computation of the following is performed at the end of the training period: (1) A_t and π_t, (2) $P(x_{t_d}|x_{t_s})$ from Eq. 8 for all possible (x_{t_s}, x_{t_d}) pairs, and (3) the heuristic for A* path-search $h(n_{t_d-k}, d_{t_d})$ for all possible d_{t_d} and $k \leq D$. In the later online stage, we compute the expected DSR value (see Eq. 9) for all (s, d) pairs for a given current location u_t, the one with maximum temporal-influence in the current time-slot, using our optimized path-tree exploration. Figure 7 in part (a) shows the run-time results of a single-threaded implementation on the mobile device for the selected subjects in our dataset. Even for the relatively expensive *pre-computations*, the median value of the run-time is below 1.5 seconds as shown on the left in this figure. Moreover, the average time for the online computation of DSR has a median value of around 100 *milli*-seconds. The variations in the run-time for pre-computations depend on the size of the training data (time-slotted days) and the number of unique locations (states) whereas the online computations are mainly affected by the number of possible (s, d) pairs for the location u_t. In a more detailed implementation, these times can be further improved by caching and re-using computed values of DSR for the unique $(s, d, \boldsymbol{u})$ tuples. In general however, we can reasonably say that our algorithm estimates DSR in real-time.

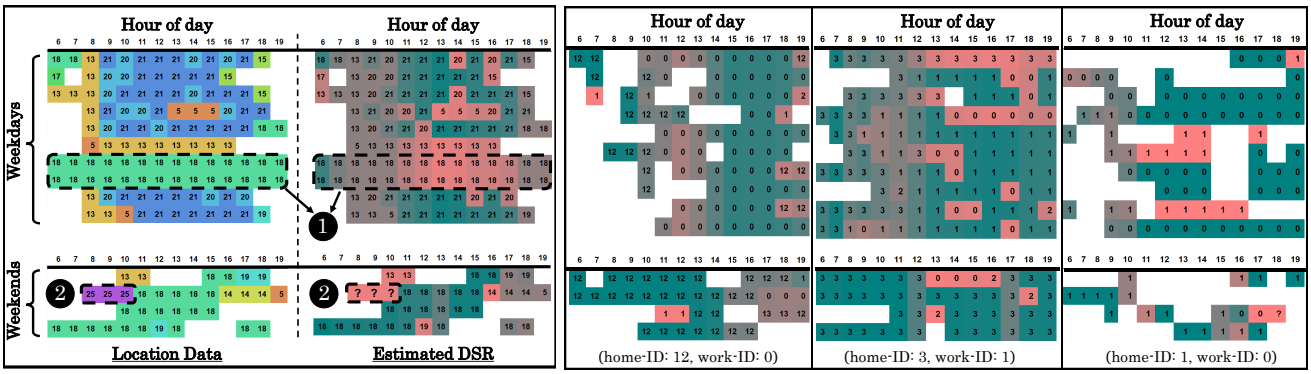
We also emphasize here that, apart from the selection of high quality training examples E_θ for policy learning (cf. Section 3), the above computations can also be reused for to serve other applications. For example, the estimated DSR values can also enable different context-aware services such as movement-novelty based automatic location updates for family safety applications. Similarly, the computed Markov transition matrices A_t are also useful for location prediction and its derived services such as active reminders.

6.2.2 Quality of Routine Signal

To access the quality of the estimated DSR signal, we first provide, in Fig. 7 (b) and (c) respectively, the density plots of DSR values for different locations against the visit counts of the subjects to these locations and their staying-durations. In general, it is evident that the locations with higher visit counts or higher staying durations, which should typically represent routine locations, are indeed associable with lower values of DSR (representing higher routine-strength).

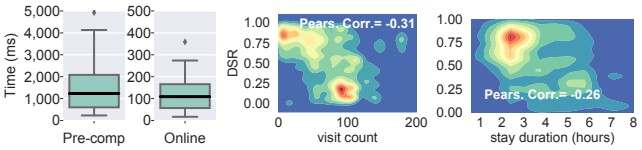
To further evaluate the intuitive value of the estimated DSR signal, Fig. 6 (a) visualizes a part of the complete time-slotted location data (on the left) of a single subject alongside the corresponding color-gradient plot of the estimated DSR values (on the right). For further ease in analyzing the variation in routine behavior, we separately plot the data for weekdays of this subject in the upper part of the figure and the weekend-days in the lower part. Moreover, for the DSR values, the colors vary between dark green, for strong routine behavior (or $DSR = 0$), and bright red, for strong out-of-routine behavior (or $DSR = 1$).

From the complete data of this particular subject, we could infer that location ID 18 was his home location and location ID 21 was his dominant work location. Looking closely at the DSR estimates in Fig. 6 (a), we can deduce that our algorithm can detect the out-of-routine behavior of the subject being at home during weekdays, as marked by ❶. Notably for this case, the DSR values gradually indicate a stronger deviation from routine behavior near the middle of the day (brighter red color) due to the otherwise strong work routine during this time. On weekends however, staying at home (ID 18) is intuitively considered routine behavior for this subject indicated by dark green color. Naturally, our algorithm also detects visits to previously unseen locations as strong out-of-routine behav-



(a) (b)

Fig. 6: (a) Side-by-side comparison of time-slotted location data and the estimated DSR values for a single subject. The DSR values vary between strong routine (dark green color) and strong out-of-routine movements (bright red color) (b) Plots of estimated DSR values for three different subjects, labeled with their corresponding home and work location-IDs.



(a) (b) (c)

Fig. 7: (a) Run-time performance results for our DLR estimation algorithm. (b) DSR vs. number of visits to different locations. (c) DSR vs. duration of visit to different locations.

ior, e.g., location ID 25 as marked by ❷ on the figure. This is realized by adding a pseudo state in the Markov chain of the subjects which has a non-zero probability of being visited due to the application of additive smoothing.

Similarly, part (b) of Fig. 7 plots the estimated DSR values for three other subjects in the dataset, while also providing their inferred home and work location-IDs. Here again, it is easy to see the intuitive trends in variation of routine strength within individual days (e.g., strong day-time work routine) and across weekdays and weekends (e.g., dominantly home-staying routine on weekends).

6.3 DSR and Information Sharing Policies

Before presenting the policy performance results, we first discuss the considered performance metrics and then describe the details regarding the compared machine-learning algorithms.

6.3.1 Utility and Privacy Metrics

For any trained sharing policy, its performance is typically evaluated in actual usage by means of three well-known metrics, namely, *under-sharing*, *over-sharing*, and *accuracy*.

Under-sharing captures the *proportion of unduly suppressed events* that the user actually wanted to share, i.e., false negatives. Thus under-sharing measures the *utility-loss* of a sharing policy. In contrast, the over-sharing metric represents the *privacy-loss*, which is defined as *the proportion of privacy-sensitive events that are erroneously shared*, i.e., false positives. Naturally, an optimal sharing policy should minimize both of these metrics. For readability, we refer to the under- and over-sharing metrics as *utility-* and *privacy-losses*, respectively, for the rest of the article.

Finally, the accuracy of a sharing policy evaluates all of its decisions into a single measure and is defined as follows:

$$Accuracy = \frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|} \tag{11}$$

where TP and FP represent true and false positives, and TN and FN represent true and false negatives, respectively.

6.3.2 Compared Policy Learning Algorithms

To enable performance comparison, we train three different algorithms that have been employed for policy learning in existing literature [4,3,2], namely, Support Vector Machines with RBF kernel (SVMs), Naive-Bayes classifier (NB), and Location-sharing Rules (LR). For

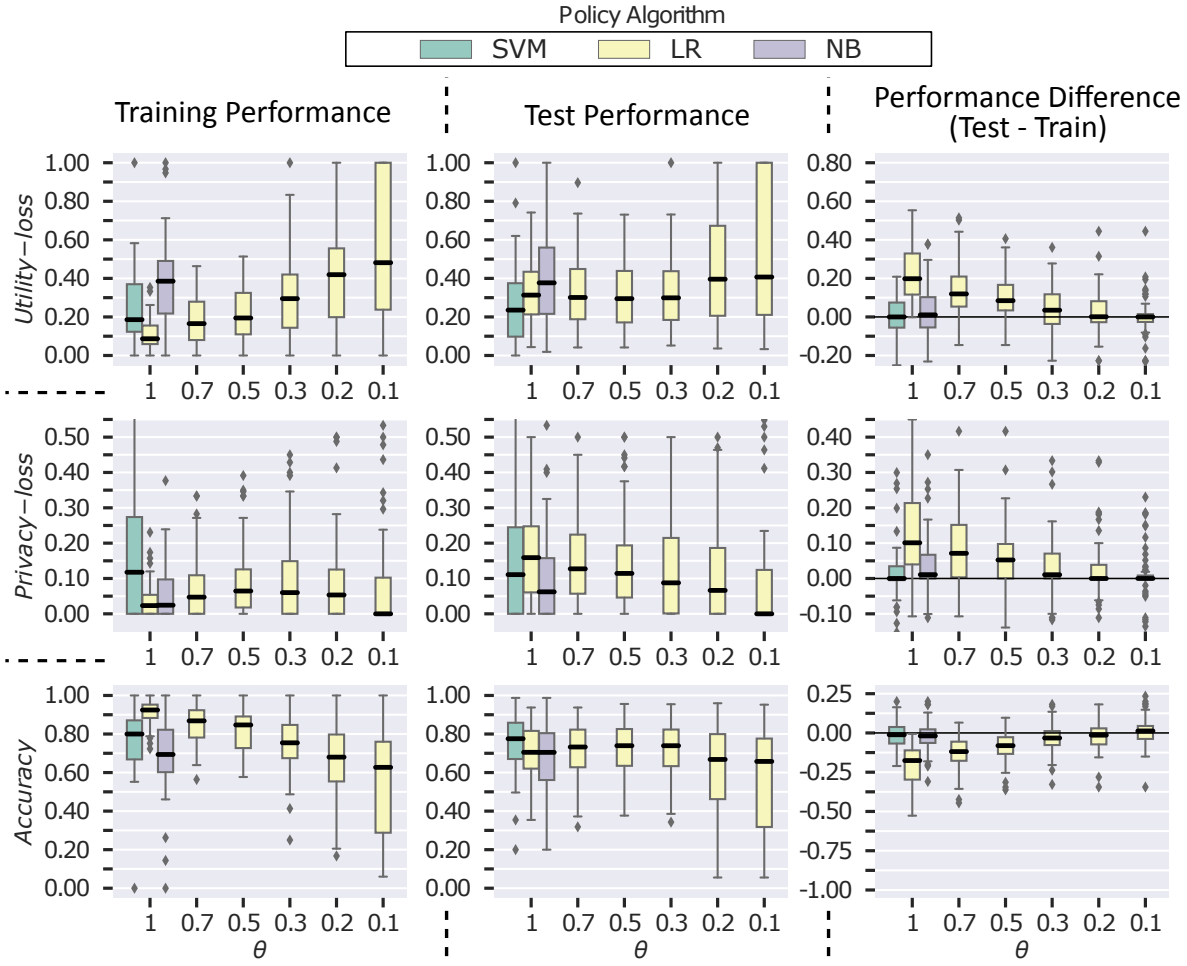


Fig. 8: Policy Performance results for $c = 1$. Column 1: training data, column 2: the test data, column 3: performance difference. Each row accounts for different performance metrics.

SVMs and NB classifiers, we use 5-fold cross validation to select the best parameters. In contrast, the LR classifier, as elaborated in Section 5, is a simplistic, parameter-free algorithm and thus does not require model selection.

Particularly, for RBF the kernel SVMs, we try 10 different values for both of its parameters (C and γ) in the interval $[10e-3, 10e3]$, resulting in a total of 100 possible models. For NB classifier, we try out 10 different values of the smoothing parameter in the interval $[0, 2]$ for learning the conditional distributions of the features given the classes. Considering the affect of 5-fold cross-validation (cf. Section 3.4), the policy-learning phase involves a total of 550 model-training operations per subject ($m*k = (100\times\text{SVM}+10\times\text{NB})*5$). In general, this number could grow further if other algorithms such as various types of decision trees are also considered for model selection or if a more fine-grained search of the best parameters is performed, thus causing an infeasible increase in the computational requirements for model selection on mobile devices.

In order to avoid bias in learning due to class imbalance (70 % positive and 30 % negative sharing events in our case), we setup the three learning algorithms to be cost-sensitive, i.e., the positive and negative events can be weighted differently. This also allows us to conduct different experiments with varying values of c, i.e., the relative cost/sensitivity of the private event. For SVMs, the cost-sensitivity is implemented by appropriately setting the class-weight parameter. For NB, the prior class probabilities are appropriately adjusted. Finally, for LR, the c value is directly used as part of the *score* function as discussed in Section 5. During model selection, we pick those SVM and NB models which yield the best accuracy, where the definition of accuracy is modified to include the affect of c by replacing $|FP|$ in denominator of Eq. 11 by $(c * |FP|)$.

We will present results for two values of c.

1. $c = 1$: the privacy loss per FP decision is considered equal to the utility loss per FN decision.
2. $c = 1000$: privacy loss per FP decision is considered 1000 times more costly.

6.4 Policy-performance Results

Recall that in order to select the set of high quality training examples E_θ (cf. Section 3.5) for policy learning, the noise-confidence η_i associated with each $e_i \in E_\theta$ must satisfy $\eta_i < \theta$. Also recall that we assume $\eta_i = DSR(e_i)$, thus implying that the training examples with higher DSR values (stronger out-of-routine behavior) are deemed more noisy.

For the case with $c = 1$, Fig. 8 presents the policy performance results for all subjects. Here, the three rows of plots correspond to results related to the utility-loss, the privacy-loss, and the accuracy metrics, respectively. As labeled on the figure, the first and the second columns of these plots show the training and the test performances of the three classifiers. The third column summarizes the difference in performance between the test and the training data. Ideally, if over-fitting is avoided, this difference for all three metrics should be zero, implying that the learned policy generalizes well in actual usage. For each of these plots, we vary the value of θ on the x-axis from 1 down to 0.1 thus selecting increasingly in-routine training examples in E_θ. Also, we compare the performance of the LR classifier trained for the different values of θ with the performance of the best SVM and NB models for $\theta = 1$, i.e., over the full training set ($E_\theta = E$).

Beginning with the test-performance (middle column), we observe that for the full training data ($\theta = 1$), the three classifiers have varying performance. Considering utility-loss first, the SVM classifier performs the best with a median value of 23 % whereas the NB classifier performs the worst with a median of 38 %. Regarding privacy-loss, the SVM and NB classifiers achieve median values of 11 % and 6 % respectively. The relatively higher value of privacy-loss for the SVM classifier is explained by its significantly lower values of utility-loss. In contrast, the LR classifier, while causing higher utility-loss (median of 31 %) than the SVM classifier, also yields the worst results for privacy-loss (median value of 16 %).

This poor performance for the LR classifier for $\theta = 1$ is caused due to over-fitting, which is evident by the correspondingly high test-to-train median performance differences of 20 %, 10 %, and 17 % for the utility-loss, the privacy-loss, and the accuracy metrics, respectively, as shown in third column of Fig. 8. However, as θ is gradually decreased from 1 to 0.3, the median value of privacy-loss for LR in the test-set reduces from 16 % to 9 %, *without increasing utility-loss*. Moreover, the third column also shows that at $\theta = 0.3$, the test and train median performance for all the three metrics differ minimally, thus indicating better generalization. Reducing θ beyond 0.3 causes further decrease of the median privacy-loss percentage for the LR policy, though this advantages comes at the cost of increased utility-loss. However, in general, these results strongly substantiate our hypothesis that training examples sampled from strong routine behavior lead to better generalization of the over-fitting prone LR policy and to a performance comparable to that of the cross-validation-enabled, complex machine learning models. For instance, at $\theta = 0.3$, the performance of the LR classifier is comparable to that of the SVM classifier at $\theta = 1$ for all the three metrics, where as at $\theta = 0.2$, its performance similarly compares to that of the NB classifier.

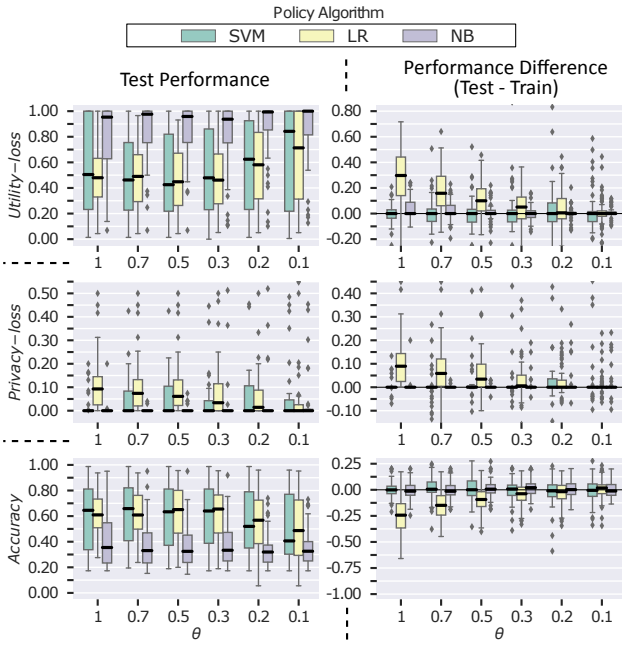


Fig. 9: Policy Performance results for $c = 1000$. Column 1: test performance, column 2: test-to-train performance difference.

In Fig. 9, we present the test performance (left) and the test-train performance-difference (right) for the three classifiers when $c = 1000$. Doing so biases the three policy learning algorithms to minimize privacy-loss at the expense of increased utility-loss. Accordingly, the SVM and the NB classifiers cause almost no privacy-loss in the test-set (see $\theta = 1$), though the NB

classifier achieves this at a significantly higher utility-loss values with a median of 95 %.

As for the LR classifier, the same trend of decrease in privacy-loss with lowering values of θ is observed. For instance, at $\theta = 0.3$, the median value of privacy-loss for the classifier reaches 4 %, with a median utility-loss of 47 % compared to that of 50 % for the SVM classifier at $\theta = 1$. Further lowering θ to 0.1 also enables a median privacy-loss of 0 % for the LR classifier, although at higher median utility-loss of 71 %. We will shortly present further results regarding this increased utility loss.

Note that in Fig. 9, we also plot the performance of SVM and NB classifiers for $\theta < 1$ alongside the values for the LR classifier. It is noticeable that for low values of θ, e.g., ≤ 0.2, the SVM classifier causes higher utility-loss than LR for approximately the same amount of privacy-loss. This also reflects in the median accuracy values of the LR classifier, which are the highest in this range of θ values. We take this result as a confirmation that highly in-routine training examples contain reliable information regarding the actual sharing behavior of users and hence over-fitting them actually improves policy performance. Since the LR classifier is prone to over-fitting in contrast to the cross-validation enabled SVM classifier, it yields a lower utility loss with comparable privacy-loss for $\theta < 0.2$.

6.4.1 Results for Variable Selection of θ per Subject

In both of the above experiments (with $c = 1$ and $c = 1000$), we noted the consistent result that privacy-loss for the LR classifier decreased with lowering values of θ from 1 to 0.3, without causing any visible increase in the utility-loss. Beyond $\theta = 0.3$ however, the decrease in privacy-loss came with significant increase in utility-loss. The main reason for this outcome was found to be the lack of enough training examples for some of the subjects. For a clearer analysis of LR classifier's performance for strongly in-routine training examples, we performed another experiment by selecting different values of θ for each subject such that at least a decent proportion of the training examples was included in E_θ.

Particularly, for any given subject, we initially determine the value of θ, denoted θ_{10}, which allowed the inclusion of the 10 percent most in-routine training examples in E_θ. Next, we set $\theta = \theta_{10}$ if $\theta_{10} \in [0.1, 0.3]$. Otherwise, θ is snapped to the nearest of boundary of this interval. In contrast to a fixed value of $\theta = 0.1$ for all subjects, this methodology attempts to include more in-routine training examples in E_θ (bounded by $\theta \leq 0.3$) for those subjects who have $\theta_{10} > 0.1$.

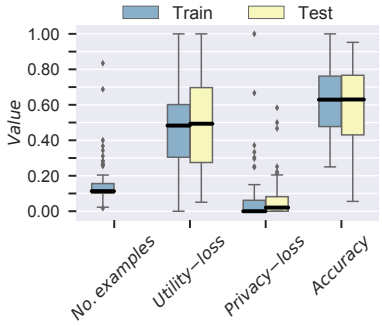


Fig. 10: LR policy performance results with flexible selection of θ for data subjects.

In Fig. 10, we present the performance of the LR policy for $c = 1000$ when θ is selected according to the above defined methodology. Note that apart from the utility-loss, privacy-loss, and the accuracy metrics, the figure also plots the distribution of the proportion of training examples in E_θ for the subjects. As expected, the median number of examples is 10 %. If we compare these results with the ones at $\theta = 0.1$ in Fig. 9, we observe a significant improvement in the LR policy's performance with slight increase in the median privacy-loss (from 0 % to 2 %). Particularly, the median utility-loss decreases remarkably by approximately 21 % (from 71 % to 50 %). This in turn shows a significant increase in the median accuracy of the LR policy by approximately 13 % (from 49 % to 62 %), making it comparable to the SVM classifiers accuracy of 65 % at $\theta = 1$. Thus flexible selection of θ, that ensures the inclusion of a small percentage of strongly in-routine samples in E_θ, does offer a simple and effective way of learning high-performance location sharing policies.

6.5 Pros and Cons of the DSR-aware LR Policy Learning

Based on the above evaluation results, we now discuss the drawbacks and advantages of using the LR classifier in conjunction with the DSR information for learning location-sharing policies.

From the policy-performance perspective, we have seen that the LR classifier can indeed perform comparably to complex classifiers such as the SVMs, when it is trained using increasingly in-routine examples. The only drawback visible from the above results is the slightly higher privacy-loss (by 2 %) for the LR classifier, when it is highly undesired, i.e., $c = 1000$. This drawback can be attributed to the algorithmic simplicity underlying the LR classifier (look-up tables), which in turn represents reduced computational complexity of

training the sharing policies. In contrast, recall that we trained 500 SVM and 50 NB models for each subject for model selection.

It is also noticeable from our results that, even for the SVM or the NB classifiers, a 0 % privacy-loss may be achieved at different values of c for different subjects. Moreover, unnecessarily high values of c may cause unwanted utility-loss such as that for the NB classifier when $c = 1000$. Thus in a real training scenario, a user may have to try a few values of c before being satisfied with the estimated policy performance. Such interactive training imposes an even stricter requirement on the response-time of the training process for practical usability. Note that trying out different values of c for typical machine-learning classification algorithms, e.g., SVMs using their class-weight parameter, requires their complete retraining, which translates to complete re-executions of the costly model selection procedure. In contrast, the LR classifier can efficiently adjust to changes in c since the underlying look-up tables remain unchanged. The value of c is only used to compute the score function during decision making (cf. Section 5). Hence the performance of the LR classifier for different values of c may be efficiently estimated on the training data.

Finally, by selecting more in-routine training examples for policy training, we define a systematic methodology for collecting training data. As shown in Fig. 10, the actual number of training examples in E_θ for learning the LR-based high-performance policies can be as low as 10 % of that in the full training-set E. Thus if a-priori estimation of policy-performance, which requires training examples over the full spectrum of user-movements (routine or non-routine), is not deemed important, then an LR-based policy may be learned over a relatively small number of strongly in-routine training examples. As is demonstrated by our evaluation results, such strongly in-routine training examples likely lead to high-performance location sharing policies. In other words, the number of in-situ questions posed to the users for collecting their policy-training data can be significantly lowered, thus reducing the amount of the required, manual user-input.

We also note that the set of in-routine training examples can also be used to bootstrap active-learning based policy-learning approaches [4]. While these approaches incrementally improve their performance over time by asking for user-input for "difficult" location requests, they initially need a basic dataset to become functional, which could be provided based on DSR-aware sampling.

7 Conclusion

In this article, we have proposed a novel algorithm for efficiently determining a user-mobility-based signal, namely, the dynamic strength of movement routine (DSR), on resource-constrained mobile devices. We not only demonstrated its efficient performance, we also showed that the estimated DSR signal can be used to improve the learning of location-sharing policies for social network users who want to automate the process of privacy-preserving information sharing with their online contacts. In particular, we showed that high-performance location-sharing policies, that exhibit low privacy and utility losses, can be learned efficiently using computationally simple algorithms on mobile devices. To achieve this, we improve training data quality by filtering out noisy training examples using the DSR signal. Overall, our results increase the usability of location-sharing policies on mobile devices.

Acknowledgements We will like to thank Mr. Murat Ünal for useful early discussions on the concepts in this article. This research is a part of project *PriLoc* (Privacy-aware Location Management) of the University of Stuttgart funded by the German Research Foundation (DFG) grant RO 1086/15-2.

References

1. Acquisti, A., Grossklags, J.: Privacy and rationality in individual decision making. IEEE Security and Privacy **3**(1), 26–33 (2005)
2. Benisch, M., Kelley, P.G., Sadeh, N., Cranor, L.F.: Capturing location-privacy preferences: quantifying accuracy and user-burden tradeoffs. Personal and Ubiquitous Computing **15**(7), 679–694 (2010)
3. Bigwood, G., Ben Abdesslem, F., Henderson, T.: Predicting location-sharing privacy preferences in social network applications. In: Proceedings of the First Workshop on recent advances in behavior prediction and pro-active pervasive computing (AwareCast) (2012)
4. Bilogrevic, I., Huguenin, K., Agir, B., Jadliwala, M., Gazaki, M., Hubaux, J.P.: A machine-learning based approach to privacy-aware information-sharing in mobile social networks. Pervasive Mob. Comput. **25**(C), 125–142 (2016)
5. Consolvo, S., Smith, I.E., Matthews, T., LaMarca, A., Tabert, J., Powledge, P.: Location disclosure to social relations: Why, when, & what people want to share. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '05, pp. 81–90. ACM, New York, NY, USA (2005)
6. Eagle, N., Pentland, A.S.: Reality mining: sensing complex social systems. Personal and ubiquitous computing **10**(4), 255–268 (2006)
7. Eagle, N., Pentland, A.S.: Eigenbehaviors: identifying structure in routine. Behavioral Ecology and Sociobiology **63**(7), 1057–1066 (2009)
8. Eagle, N., Quinn, J.A., Clauset, A.: Methodologies for continuous cellular tower data analysis. In: International Conference on Pervasive Computing, pp. 342–353. Springer (2009)

9. Ekroot, L., Cover, T.M.: The entropy of markov trajectories. IEEE Transactions on Information Theory **39**(4), 1418–1421 (1993)
10. Fang, L., Lefevre, K.: Privacy wizards for social networking sites. In: Proceedings of the 19th International Conference on World Wide Web, WWW '10, pp. 351–360. ACM, New York, NY, USA (2010)
11. Farrahi, K., Gatica-Perez, D.: What did you do today?: discovering daily routines from large-scale mobile data. In: Proceedings of the 16th ACM international conference on Multimedia, pp. 849–852. ACM (2008)
12. Forney, G.D.: The viterbi algorithm. Proceedings of the IEEE **61**(3), 268–278 (1973)
13. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer (2003)
14. Kafsi, M., Grossglauser, M., Thiran, P.: The entropy of conditional markov trajectories. IEEE Trans. Information Theory **59**(9), 5577–5583 (2013)
15. Kafsi, M., Grossglauser, M., Thiran, P.: Traveling salesman in reverse: Conditional markov entropy for trajectory segmentation. In: Data Mining (ICDM), 2015 IEEE International Conference on, pp. 201–210. IEEE (2015)
16. Kelley, P.G., Hankes Drielsma, P., Sadeh, N., Cranor, L.F.: User-controllable learning of security and privacy policies. In: Proceedings of the 1st ACM Workshop on Workshop on AISec, AISec '08, pp. 11–18. ACM, New York, NY, USA (2008)
17. Lipford, H.R., Besmer, A., Watson, J.: Understanding privacy settings in facebook with an audience view. In: Proceedings of the 1st Conference on Usability, Psychology, and Security, UPSEC'08, pp. 2:1–2:8. USENIX Association, Berkeley, CA, USA (2008)
18. Mcinerney, J., Stein, S., Rogers, A., Jennings, N.R.: Breaking the habit: Measuring and predicting departures from routine in individual human mobility. Pervasive Mob. Comput. **9**(6), 808–822 (2013)
19. Patil, S., Lai, J.: Who gets to know what when: Configuring privacy permissions in an awareness application. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '05, pp. 101–110. ACM, New York, NY, USA (2005)
20. Patil, S., Norcie, G., Kapadia, A., Lee, A.J.: Reasons, rewards, regrets: Privacy considerations in location sharing as an interactive practice. In: Proceedings of the Eighth Symposium on Usable Privacy and Security (SOUPS '12), pp. 5:1–5:15. ACM, New York, NY, USA (2012)
21. Ravichandran, R., Benisch, M., Kelley, P.G., Sadeh, N.M.: Capturing social networking privacy preferences. In: Proceedings of the 9th International Symposium on Privacy Enhancing Technologies, PETS '09, pp. 1–18. Springer-Verlag, Berlin, Heidelberg (2009)
22. Riaz, Z., Dürr, F., Rothermel, K.: Understanding vulnerabilities of location privacy mechanisms against mobility prediction attacks. In: Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, Melbourne, Australia, November 7-10, 2017., pp. 252–261 (2017)
23. Roberts, S.G., Dunbar, R.I.: Communication in social networks: Effects of kinship, network size, and emotional closeness. Personal Relationships **18**(3), 439–452 (2011)
24. Sadeh, N., Hong, J., Cranor, L., Fette, I., Kelley, P., Prabaker, M., Rao, J.: Understanding and capturing people's privacy policies in a mobile social networking application. Personal Ubiquitous Comput. **13**(6), 401–412 (2009)
25. Song, C., Qu, Z., Blumm, N., Barabási, A.L.: Limits of predictability in human mobility. Science **327**(5968), 1018–1021 (2010)
26. Toch, E., Cranshaw, J., Drielsma, P.H., Tsai, J.Y., Kelley, P.G., Springfield, J., Cranor, L., Hong, J., Sadeh, N.: Empirical models of privacy in location sharing. In: Proceedings of the 12th ACM International Conference on Ubiquitous Computing, UbiComp '10, pp. 129–138. ACM, New York, NY, USA (2010)
27. Wellman, B., Wortley, S.: Different strokes from different folks: Community ties and social support. American journal of Sociology **96**(3), 558–588 (1990)
28. Wiese, J., Kelley, P.G., Cranor, L.F., Dabbish, L., Hong, J.I., Zimmerman, J.: Are you close with me? are you nearby?: Investigating social groups, closeness, and willingness to share. In: Proceedings of the 13th International Conference on Ubiquitous Computing, UbiComp '11, pp. 197–206. ACM, New York, NY, USA (2011)