The Thirty-First International Florida
Artificial Intelligence Research Society Conference (FLAIRS-31)

Using A Personalized Anomaly Detection Approach with Machine Learning to Detect Stolen Phones

Huizhong Hu, Philip K. Chan
Florida Institute of Technology, Melbourne, FL 32901
hhu2013@my.fit.edu , pkc@cs.fit.edu

Abstract

We devise an anomaly detection system that detects stolen phones. In this system, we use a mining algorithm to extract sequential patterns from a user's past behavior to construct a personalized model. We then put forward scoring functions and threshold setting strategies to detect stealing events. We evaluate our approach with a data set from the MIT Reality Mining project. Experimental results indicate that our approach can detect 87% of simulated stealing events with an average false positive rate of 0.9%.

Introduction

Smartphones have become ever more functional and users are more dependent on their smartphones. If your phone is lost or stolen, a nightmare will soon begin. One will worry about not only losing the phone hardware, but also losing personal information, which might lead to identity theft or worse consequences. With the global growth in the usage of smartphones, phone theft has become an increasingly significant problem. In the United States, 113 phones are lost or stolen every minute. According to the U.S. Federal Communications Commission, nearly one third of robberies involve smartphones. In 2012, smartphone crimes cost 1.6 million Americans about 30 billion dollars. The figure almost doubled in 2013—3 million Americans became victims of smartphone crimes. (US-FCC).

"How to detect a stolen phone?" has become a difficult but urgent issue. To solve this problem, the U.S. government and the Mexican government have developed some countermeasures. In 2012, a number of communication companies including AT&T collaborated and built a central database of stolen smartphones. Every time a mobile phone is reported to be missing and registered in the database, its unique serial number will be recorded. Then the mobile operator can block any connection to that number. Apple and Samsung use a different way to handle this issue. They provide a service that sends back the phone's location to the owner or restores factory settings of the phone.

Solutions above require the phone owner's awareness of losing the phone. Is there a way to alert the owner more promptly as soon as the stealing happens? We resort to machine learning to realize self detection. Based on location data collected from the phone sensors, we use a pattern mining approach to construct a personalized model of customary behaviors. Comparing patterns of the model with those of the current behavior, we can generate a score for detecting anomalies. Our achievements include:

- Building a personal profile of the user's mobility with a pattern mining algorithm,
- Methods for calculating anomaly scores and thresholds for detecting anomalies, and
- Experimental results indicating that our approach can detect 87% of the anomalous behavior with only 0.9% false positive rate.

Related Work

To detect suspicious behavior under WLAN connections, (Tandon and Chan 2009) applied an algorithm for temporal location anomaly detection to learn the distributions of location probability, specifically by using a combination of sequence of time and location. To discern anomalies, a modified Markov model was employed to calculate anomalous scores that represent differences and similarities of summarized location probability distributions.

To track lost phones, (Zhang et al 2010) used a one-hour record of Cell Tower ID as location data and generated Cell ID Entropy, which represented how fast or how far the cell phone moves within a certain period of time. A Feed Forward Neural Network used hourly call counts to detect whether the phone is statically lost (e.g. left it in a library), dynamically lost (e.g. left it in a moving taxi), or being normally used. Farrahi et al. (2010) presented an approach for large-scale unsupervised learning and predicting people's routines through the joint modeling of human

Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

locations and proximity interactions by using the Latent Dirichlet Allocation probabilistic topic model.

Lu et al. (2014) used GPS data and application data to predict which applications will be used next. First, they preprocessed location data and transformed the GPS geographic locations into semantic locations. Then Lu et al utilized a density-based clustering algorithm to find the motion path and then built a Mobile App Sequential Pattern Tree to represent the correlations between locations and applications with path data. Liao et al. (2012) combined the launch time and previous application data to predict and advise applications. The App Usage Predictor component, based on Chebyshev's inequality, provides a probability-based scoring function. Liao et al (2013) extracted three features from App Usage Predictor and calculated the usage probability of each app. The Global Usage feature gives a probability statistic of the app, derived from the total number of times that the app is used during the whole time. The Temporal Usage feature gives another probability statistic of the app usage within a period of time. The Periodical Usage feature indicates usage habit, which measures how frequently the app is in use. At the end, the Min Entropy Selection counts the entropy of each feature and selects the best one for prediction. Shin et al. (2012) used more data from smartphone sensors to perform a comprehensive analysis of the context related to mobile app use, and built prediction models to calculate the probability of an app in different contexts.

In this study, instead of generating only one attribute that describes phone loss in locational facts (Zhang 2010), we analyze behavioral patterns that detect phone loss. Instead of reacting to loss events three hours later, our approach aims to predict a loss within an hour.

Approach

Our goal is to use machine learning algorithms to personalize user behavior in order to automatically detect phone loss and protect owners' property and privacy. Our approach contains two parts: behavior learning and anomaly detection. To learn user behaviors, we use the SPAM pattern mining algorithm (Ayres et al. 2002) to identify a behavior pattern set from raw location data, and then we merge and process them into a personalized model set. This is different from the work by Farrahi et al. (2010) which uses unsupervised learning to cluster a user's past behavior. To detect anomalies, we associate the input data with the current behavior. After extracting behavior patterns, we find correlations between the current pattern and the recorded personalized profile, and derive an anomaly score. Based on previous data we determine a threshold for the anomaly score. Patterns with scores above the threshold are considered anomalous. Due to space limitations, some details are left out. More details are in (Hu 2015).

Data Preprocessing

The raw data contains time record and location information, that is, the area ID and the cell tower ID, which connect the phone. We remove data that are identified as non-compliant (such as no signal, only time record, or location info with either only area ID or only tower ID). We categorize these data into a date ordered format to build seven daily models, from Monday through Sunday. The underlying reason is that most people schedule weekly and they usually repeat similar schedules every seven days.

Furthermore, we use a modified version of the sliding-window to separate one-day dataset into 24 hourly subsets. The hourly subsets are used to build individual hourly models that describe human daily behavior in each time slot. Moreover, this window covers a two-hour time period, including one current hour, one half hour before, and one half hour after the current hour.

Additionally, the main reason we set one hour as the basic unit is that we want to detect a stolen phone within one hour after the phone is stolen. Naturally, the earlier one detects a stolen phone, the more possible one can recover it. In this sense, late detection would be meaningless. Secondly, we use two half-hour shifted datasets because one cyclical activity may not always occur within exactly that hour, so we want to relax the data range.

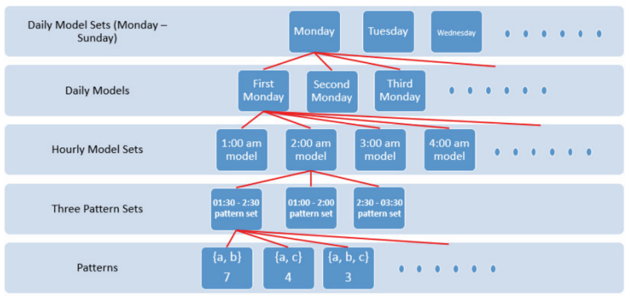

Figure 1 — The whole structure of the Personalized Model.

Behavior Learning

The structure of our personalized model is shown in Figure 1. We explain each level in bottom-up manner. The first level is the pattern level, which contains a number of patterns and each pattern contains a sequence and its frequency, which are identified by the SPAM algorithm (Ayres et al. 2002). Given sequences of itemsets, the SPAM algorithm identifies frequent sequential itemsets, which might include gaps as patterns. The second level is the pattern set level, which we merge patterns in the three one-hour datasets. Hourly model sets are level 3, which has 24 hourly model sets and each model set is constructed from three pattern sets from level 2. Furthermore, level 4 is daily models generated by multiple sets of daily data. The last level is daily model sets which contain seven model sets through the whole week from Monday to Sunday. Each model set contains multiple daily models through the whole data interval. For scoring purposes, we merge all daily models into one daily model and discuss it specifically in the next section (We do not merge them in the first place because we apply the K-Fold Cross Validation approach in all daily models to set thresholds). Our personalized model is composed of all of these seven models.

Scoring Functions of Behaviors for Anomaly Detection

The previous section explains how a personalized model is constructed. In this section we demonstrate how the model is used to detect anomalous behaviors. Given a personalized model, via a scoring function, we calculate how similar/dissimilar the current behavior is to the model. Here we suggest two scoring methods as follows.

Scoring Similarity in Behaviors

One way to score a new behavior is to evaluate the similarity between the new and previous behaviors. More specifically, we calculate the score of similarity between the test pattern sets and the corresponding hourly model in the personalized model.

Firstly, we merge all corresponding model sets from the personalized model into one model set. Later we use the merged personalized model to calculate the score. To use the Monday model set (level 5) as an example, we merge all Monday daily models (level 4) into one Monday model. Additionally, we merge hourly model sets (level 3) correspondingly, and the three pattern sets (level 2) into one model. On level 1, if two patterns are the same, we sum up all frequencies; otherwise we simply copy the pattern and frequency into the merged pattern set and all the frequencies are divided by the number of days in the end.

To score new data in the one-hour time period, firstly, we extract a pattern set by applying the SPAM algorithm to the new data. Then we compare this pattern set with the merged personalized model. Here we discuss three cases: Case 1 is when a pattern appears in both the model and the test pattern set. Case 2 is when a pattern occurs only in the model (hence absent from the test pattern set). Case 3 is when a pattern appears in only the test pattern set (hence absent from the model).

We define the similarity scoring function *SF*1 as below:

$$O_i = \min\left(F_p(i),\ F_t(i)\right) \qquad (1)$$

$$S_o = \sum_{i=1}^{number\ of\ \ Patter\ in\ Case1} O_i \qquad (2)$$

$$SF1 = \frac{\frac{S_o}{S_p} + \frac{S_o}{S_t}}{2} \qquad (3)$$

$F_p(i)$ is the frequency of pattern *i* in the personalized model and $F_t(i)$ is the frequency of pattern *i* in the test pattern set, so O_i means the frequency of overlapping pattern *i*. S_o is the sum of all overlapped pattern frequency values. Cases 2 and 3 are not considered because the overlapped pattern frequency is always zero. S_p is the total frequency value in the personalized model; similarly, S_t is the total frequency value in the test pattern set. Consequently, *SF*1 represents the percentage of overlapped pattern frequency in the training and test sets, and then the value is normalized.

Scoring Difference in Behaviors

Another way to score a new behavior is to evaluate the difference between the new and previous behaviors. It is almost the opposite of similarity but calculated by a different set of formulas, which we call *SF*2. We calculate the score of difference between the test pattern sets and the corresponding hourly model in the personalized model. We define *SF*2 as below:

$$D_i = |F_p(\text{i}) - F_t(i)| \qquad (4)$$

$$S_D = \sum_{i=1}^{number\ of\ \ Patter\ in\ Case1,2\ \text{and}\ 3} D_i \qquad (5)$$

$$SF2 = \frac{S_D}{S_p + S_t} \qquad (6)$$

D_i is the difference in frequency of pattern *i* between the personalized model and the test set. We subtract the test pattern frequency $F_t(\text{i})$ from the training pattern frequency $F_p(\text{i})$, for pattern i, and then take the absolute value. Moreover, the difference of pattern frequency for Case 2 and 3 is its frequency subtracted by zero which equals to itself. S_p and S_t are defined the same as in the previous section.

Score Thresholds of Anomaly

After a score for the current behavior is obtained, we need to mark thresholds to identify if the current behavior is anomalous or not. This section discusses two scenarios. In the first scenario, only data from the user (negative examples) are available for training. In our case, this scenario proposes an anomaly detection problem. In the second scenario, in addition to data from the user (negative examples), data from other users (positive examples) are also available for training.

Anomaly detection (with only user data during training)

In this scenario, the user's smartphone can collect the user's behavioral data, build a personalized model, and detect anomalous behaviors. That is, user data need not be shared to any other entities. To identify anomaly, we use K-Fold Cross Validation to find a threshold. In the personalized model, one of the daily model sets (level 5) has multiple daily models (level 4). Additionally, if there are k models, we merge k-1 models as a training mode by using the scoring strategy that we introduce in the previous section to score the remaining model. After doing k times of iterations with a different k, we can get k score lists. More specifically, each score list has 24 scores since each daily model has 24 models (level 3), which correspond to 24 hours a day. Now we take the 24 lowest scores across all score lists as our threshold list (the lower score is, the fewer similarities exist between training and test). We refer to it as T_{u_i} (0 <= i <= 24), which represents each hour's threshold. Therefore, when a new behavior comes, we calculate the score by merging all models in the personalized model and using it as a training model. Once the score is lower than the corresponding threshold, we report it as an anomalous event. We call this Strategy 1. Furthermore, we devise a refined strategy that adjusts the threshold to reduce false positive rate, and we call it Strategy 2. Instead of using the fixed threshold, we reduce threshold from T_{u_i}‘ (0 <= i <= 24) to T_{min_i}’ (0 <= i <= 24) every time it detects a new normal activity as stolen.

Using behaviors of other users to help determine thresholds

In this subsection, we explore the scenario when behavior data from other users are also available. In a real-world setting, behavior models from participating users can be uploaded and stored in an external central database/server. To preserve the privacy of participating users, the central server determines the score thresholds without sending behavior models to any user. Since behaviors of other users (potentially thieves) are not desirable, we would like to determine another score threshold (T_{o_i}) for behavior of other users. Figure 2 illustrates the relationship between the two thresholds. On the left panel of Figure 2, only the user data are used and T_{u_i} is identified. On the right panel of Figure 2, we also identify T_{o_i} from other users’ data. Behaviors between the two thresholds are in the “gray area”, and will be classified as “unknown”--neither the user (normal) nor others (anomalous).

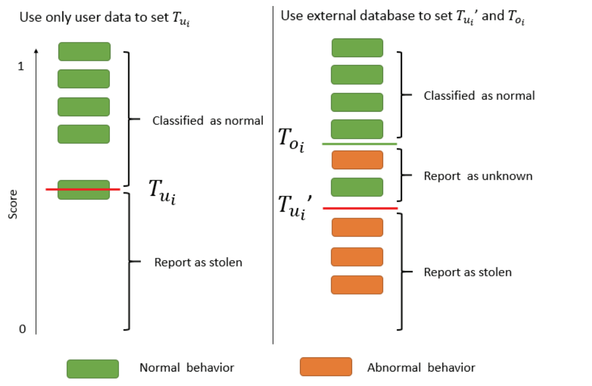

Figure 2 —Setting thresholds with data from the user, and with data from both the user and other users in an external database.

More specifically, the external dataset with behavior from other users must have enough user data and then be used in a similar way to set T_{o_i}. Firstly, we find the highest score for other users from the database (behaviors from the other users are considered anomalous) and then set a threshold between this and the closest higher user score. Moreover, as is shown in Figure 2, we set T_{u_i}’ to be not the lowest user score, but between this user score and the closest lower validation score. Then we predict events, of which the score is over T_{o_i}, as a normal activity, those of which the score is between T_{o_i} and T_{u_i}’ as unknown behavior, and those below will be reported as stolen. This is Strategy 3.

Dynamically adjusting the thresholds

To dynamically adjust the threshold during detection, similar to Strategy 2, we use the false positives to decrease T_{u_i}, rendering Strategy 4.

Table 1 summarizes our four strategies. They all use user data only during the training process, but Strategies 1 and 2 only use user data to set the threshold, while Strategies 3 and 4 use both user and other data to set the thresholds. Moreover, Strategies 1 and 3 use static thresholds during detection, while Strategies 2 and 4 adjust thresholds dynamically during detection to reduce false alarms.

Table 1 —Overview of Strategies on setting and adjusting the threshold.

	No adjustment during test	Adjust during test
Setting user data only	Strategy 1	Strategy 2
Setting with user and other data	Strategy 3	Strategy 4

Experimental Evaluation

The dataset we use is sourced from Reality Mining (Eagle et al. 2009), a project conducted at the MIT Media Laboratory. The data were collected from the smartphones of 94 individuals working or studying at a university from September 2004 to June 2005. We are provided with call logs, Bluetooth devices’ connection data, cell tower IDs, application usage, and status of mobile phones. Of these 94 subjects, 68 were working or studying in the same general location on the main campus. The other 26 subjects were new students from the business school in the university. The dataset contains 90% graduate students and 10% staff.

For each user, after preprocessing, we have three types of data: (1) the cell tower transition time and cell tower ID pair (e.g., 26-Jan-2005 16:42:35, 24127.0011), which represents location information, (2) log time and application name pair (e.g., 26-Jan-2005 16:39:51, Menu), and (3) time and activity pair (e.g., 26-Jan-2005 16:57:30, 1), which uses 1 and 0 to represent whether the phone is being used or not. Since Type 2 and 3 are quite sparse in the dataset, we only use the first type of data in this study. That is, we focus on the spatio-temporal behavior of the users. To ensure sufficient data for building and evaluating our models, we removed users with fewer than 120 days of data in the sampling period. Data for 42 users remained valid. However, some of the users have long periods of no activities.

To evaluate our system, we use several criteria:

1. True Positive Rate (TPR)

$$TPR = \frac{TP}{TP+FN} \tag{7}$$

2. False Positive Rate (TPR)

$$FPR = \frac{FP}{FP+TN} \quad (8)$$

3. Area Under the ROC Curve (AUC)

TP is denoted as True Positive and is counted only when the phone is defined as stolen and the algorithm predicts correctly. FN means False Negative and is counted when the algorithm cannot detect stolen status. Similarly, FP represents False Positive and is counted when the algorithm gives an false alarm, and TN means True Negative and is counted when the algorithm recognizes the owner's identity. A summary is in Figure 3.

When the incoming user's behavior score is between T_{o_i} and T_{u_i}', the behavior is classified as UN (unknown negative) and when the other users' behavior score between these two thresholds is classified as UP (unknown positive). Since our algorithms can choose not to make prediction, we modify TPR and FPR to include unknown predictions in Equation 9 and 10. Additionally, Equation 11 and 12 calculate the unknown positive and negative rates.

All modified formulas are shown below:

$$TPR = \frac{TP}{TP+FN+UP} \quad (9)$$

$$FPR = \frac{FP}{FP+TN+UN} \quad (10)$$

Unknown Positive Rate (UPR)

$$UPR = \frac{UP}{TP+FN+UP} \quad (11)$$

Unknown Negative Rate (UNR)

$$UNR = \frac{UN}{FP+TN+UN} \quad (12)$$

AUC is the area under the Receiver Operating Characteristic (ROC) Curve, the X axis of ROC represents FPR, and the Y axis represents TPR. We use different thresholds to plot the ROC and calculate the AUC for that curve.

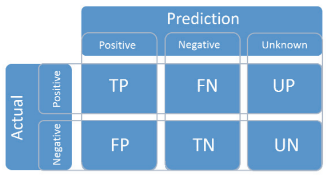

Figure 3 —Classifications table with unknown prediction

Results for Anomaly Detection Algorithms

For anomaly detection, the training set has only data from the phone owner (negative class). We first compare the outputs of our scoring functions SF1 and SF2. We then compare the outputs of SF1 and SF2 with that of Hidden Markov Model, which estimates the probability of sequence. Finally, we compare the outputs of using Strategy 1 and Strategy 2. To compare the outputs of the scoring functions with that of HMM, we use AUC, which provides a single measurement for comparison. Since high false positive rates could be annoying to users and might cause the user to ignore the alerts, we measure AUC up to 1% FPR. That is, we measure the performance of the algorithms with FPR at 1% or below. Table 2 shows the AUC values for the three algorithms. We observe that SF1 and SF2 are similar, but both are more than twice as effective as HMM-R with FPR under 1%.

Table 2 —AUC for SF1, SF2 and HMM (FPR under 1%)

	SF1	SF2	HMM
AUC	0.008005	0.008007	0.003235

Table 3 shows the average TPR and FPR of Strategy 1 and 2. As we expect, Strategy 2, with an adjustable threshold, is more effective than Strategy 1 with a fixed threshold.

Table 3 —Average TPR and FPR for all users

	TPR	FPR
Strategy 1	**88%**	7%
Strategy 2	87%	**5%**

Results for Classification

While anomaly detection algorithms uses a training set that contains data from only the phone owner (negative class), classification algorithms uses a training set that contains data from both the phone owner (negative class) and other users (positive class). In this section, we evaluate Strategy 3 that uses a fixed threshold and Strategy 4 that uses an adjustable threshold. Both strategies use two thresholds and output an unknown prediction when the score is between the two thresholds. Table 4 shows the results for each strategy. Comparing Table 4 with Table 3, we observe that the TPR rates are similar for anomaly detection (Strategies 1 and 2) and classification (Strategies 3 and 4), but the FPR rates are significantly lower for classification. The improvement is mainly due to the availability of data from the positive class (other users), which helps set a second threshold to detect "thieves". Strategies 3 and 4 have an UPR rate of about 9%, which means 9% of the positives are predicted as unknown.

Table 4 —Strategy 3 versus Strategy 4

	TPR	FPR	UPR	UNR
Strategy 3	**87.5%**	1.0%	9.8%	**46.1%**
Strategy 4	87.3%	**0.9%**	**9.4%**	46.3%

Though Strategy 4 achieves a 0.9% FPR which represents roughly 1 false alarm every five days, we would like to investigate the best-case scenario when the FPR is 0%. That is, the phone owner would not experience any false alerts. We manually find the threshold value, T_{min}, that achieves a 0% FPR and measure the TPR. Table 5 shows a TPR of 86.1%, which is only about 1% lower than Strategies 3 and 4. Moreover, because T_{min} is the best threshold we can achieve with respect to FPR, we can compare $T_{\min}$ (best-case scenario) with T_{u} (in Strategy 2) and $T_u{}'$ (in Strategy 4) to see how close they are to the best of circumstances. In Table 5, both strategies find thresholds that are close to the best threshold—1.6% higher for Strategy 2 and 4.1% higher in Strategy 4.

Table 5 —Performance of using $\boldsymbol{T_{min_i}}$

Tu / Tmin	1.016
Tu' / Tmin	**1.041**
TPR of Tmin	86.1%
FPR of Tmin	0%

Comparison with Classification Algorithms

We would like to compare our approach in Strategies 3 and 4 with other machine learning algorithms for classification, in which positive and negative examples are both available for training. Particularly, we use Decision Trees (C4.5), Decision Tree with Rule Post-pruning (C4.5 –P), Random Forests (RF), Artificial Neural Networks. We extract four features for each data record. Feature 1 and 2 are number of patterns that are common to both the user and the other users. Feature 3 is the number of patterns that occur only to the user. Feature 4 is the number of patterns that occur only to the other users.

Table 6 shows the results of our approach and other machine learning algorithms. Other learning algorithms have a higher TPR, but also a higher FPR than our two algorithms. As we discussed before, we prefer the FPR to be 1% or lower since a user might get annoyed by frequent false alerts and disable the system. Also the likelihood of getting a phone stolen is generally not high. Thus, a low FPR is more desirable than a high TPR in practice. Our two algorithms have the two lowest FPRs compared with the other algorithms, and still achieve more than 87% TPR.

Table 6 —Performance of different classification algs.

	TPR	FPR
SF1	87.9%	**0.9%**
SF2	87.6%	1.0%
C4.5	92.0%	7.6%
C4.5 - P	92.0%	8.0%
RF	**93.0%**	6.5%
ANN	89.0%	10.0%

Conclusions

In this paper, we propose an approach to detect stolen phones. First, we preprocess data into hourly subsets. Second, we apply a modified sequential pattern mining algorithm to extract sequential behavioral patterns from the data. Third, from those patterns generated from hourly data, we construct a personalized model with five levels of abstractions. To analyze the similarities between a current pattern and a pattern in the model, we propose scoring functions to calculate how similar a new behavior is to the past behavior. We use user data and K-Fold Cross validation to find the threshold and adjust it when alerts are confirmed to be false during the detection phase. Alternatively, we can add external data from other users to find the threshold and similarly adjust it when alerts are confirmed to be false. Our experimental results indicate that our approach achieves 87.9% TPR with 0.9% FPR on detecting stolen phones. Moreover, because the training time of our algorithm for one year's data is less than 20 seconds and test time is far less than one second, our system can easily run on mobile phones. Since there is no other existing stolen phone self-detection system, this is the most viable approach up to date.

References

Ayres, J; Flannick, J; Gehrke, J.; Yiu, T 2002. Sequential pattern mining using a bitmap representation. *Proceedings of the eighth ACM SIGKDD*, 429-435.

Eagle, N.; Pentland, A.; and Lazer., D. 2009. Inferring Social Network Structure using Mobile Phone Data. *Proc. National Academy of Sciences (PNAS)*, 15274-15278

Farrahi, K.; Gatica-Perez, D. 2010. Probabilistic Mining of Socio-Geographic Routines from Mobile Phone Data. *IEEE Journal of Selected Topics in Signal Processing*. 2010, Pages 745 – 755.

Hu, H. 2015. Using a Personalized Machine Learning Approach to Detect Stolen Phone. MS Thesis, Florida Tech.

Liao, Z. X.; Rey P; Lei, Shen, T. J.; Li, S. C.; Peng, W. C. 2012, AppNow: Predicting Usages of Mobile Applications on Smart Phones. *Conference on Technologies and Applications of AI.*, 300 – 303.

Liao, Z. X.; Pan, Y. C.; Peng, W. C.; Lei, P. R. 2013. On Mining Mobile Apps Usage Behavior for Predicting Apps Usage in Smartphones. *Proc 22nd ACM intl conf on information & knowledge management,* 609 – 618.

Lu, E.; Lin, Y.; and Ciou, J-B. 2014. Mining mobile application sequential patterns for usage prediction. *IEEE International Conference on Granular Computing*, 185 – 190.

Shin, C.; Hong, J. H.; Anind, A D. 2012. Understanding and Prediction of Mobile Application Usage for Smart Phones. *Proceedings of ACM Conference on ubiquitous computing*, 173 – 182.

Tandon, G.; and Chan, P. 2009. Tracking User Mobility to Detect Suspicious Behavior. *Proc. SIAM Intl. Conf. on Data Mining,* 871-882.

Zhang, C.; Fisher, R.; Wei, J 2010. I Want to Go Home: Empowering the Lost Mobile Device. *IEEE 6th Intl Conf Wireless and Mobile Computing, Networking and Communications*, 64 - 70.