Quantifying the Uncertainty of Next-Place Predictions

Paul Baumann
TU Dresden
Dresden, Germany
paul.baumann@tu-dresden.de

Marc Langheinrich
Università della Svizzera Italiana (USI)
Lugano, Switzerland
marc.langheinrich@usi.ch

Anind K. Dey
Carnegie Mellon University
Pittsburgh, PA, USA
anind@cs.cmu.edu

Silvia Santini
Università della Svizzera Italiana (USI)
Lugano, Switzerland
silvia.santini@usi.ch

ABSTRACT

Context-aware systems use predictions about a user's future state (e.g., next movements, actions, or needs) in order to seamlessly trigger the display of information or the execution of services. Predictions, however, always have an associated uncertainty that, when above a certain threshold, should prevent a system from taking action due to the risk of "getting it wrong". In this work, we present a context-dependent "level of trust" estimator that is able to determine whether a prediction should be trusted – and thus used to trigger an action – or not. Our estimator relies on ensemble learning to adapt across different users and application scenarios. We demonstrate its performance in the context of a popular problem – next-place prediction – and show how it outperforms existing approaches. We also report on the results of a survey that investigated user attitudes towards mobile-phone-based personal assistants and their ability to trigger actions in response to predictions. While users appreciated such assistants, they had substantially different tolerance thresholds with respect to prediction errors depending on the use case. This further motivates the need for a context-dependent level of trust estimator.

1. INTRODUCTION

Mobile Personal Assistants (MPAs) like Apple's Siri, Google Now or Windows' Cortana promise to materialize Salber et al.'s vision of "*a system that anticipates the user's intent and performs the task for her*" [38]. Knowing the current position of the user, for instance, an MPA can provide her with answers to as-of-yet unasked questions ("When is the next bus leaving for home from this stop?") or bring up reminders ("Leave now to catch your 2 p.m. flight!").

Today, this new wave of "anticipatory computing" [31] or "predictive intelligence" [15], as it has been called in the press, still largely focuses on proactively providing information. However, recent research is already looking into using such predictions to execute *actions*, e.g., controlling a home heating system [40, 25] (i.e., turning the heating on or off) according to the expected presence of the user during the day. As MPAs move from simply displaying information to actually executing actions on behalf of the user, the cost of "getting it wrong" rises substantially. Being presented with unnecessary or wrong information can annoy the user but causes no further harm. Instead, a wrong operation of the heating system might cause user discomfort (if the heaters are wrongly kept off) or even have monetary costs (if the heaters are activated unnecessarily).

We therefore argue that an MPA should execute actions on behalf of the user only when it can predict her needs "trustful enough". However, quantifying the "level of trust" of the predictions of an MPA is challenging and it depends both on the application scenario and the user's preferences and attitude.

In this paper, we focus on the specific scenario in which MPAs trigger actions following predictions about the next place visited by the user. We thereby use Kim et al.'s definition of a *relevant place* as a location "*where the user spends a substantial amount of time and/or visits frequently*" [21]. The next-place prediction problem has been investigated extensively and a large number of next-place predictors is available in the literature [43, 12, 39, 33, 44]. Significant less attention has instead been given to understanding how to quantify and deal with the uncertainty inherent in next-place predictions.

Next-place predictors are often implemented using common classifiers – e.g., Support Vector Machine (SVM) or k-Nearest Neighbor (k-NN) – or combinations thereof [12, 24, 6]. Given some input data, classifiers provide as output the most likely class to which the input belongs to, whereas each relevant place corresponds to a class. Along with their estimate, classifiers also usually provide a *class membership probability* [49, 50] for each prediction. This probability – similar to Chow's *prediction reliability* [10] – indicates how probable it is that the current input actually belongs to the predicted class, i.e., the probability that the current prediction is actually correct. A straightforward strategy to determine whether the prediction is trustful enough consists in setting a threshold and trigger an action only if the class membership probability of the prediction exceeds the threshold. This corresponds to interposing a binary classifier between the next-place predictor and the actual MPA, as illustrated schematically in Figure 1a. The MPA utilizes the current prediction only if the output of the binary classifier indicates that the current prediction is trustful enough.

Relying solely on the class membership probability to decide

MobiCASE 2016, November 30-December 01, Cambridge, Great Britain

DOI 10.4108/eai.30-11-2016.2267147

whether to accept or reject a prediction might however be very prone to errors. It has been shown in the literature that many classifiers provide inaccurate estimates of class membership probabilities [49, 50, 36]. Techniques to improve these probabilities – e.g., classifiers' calibration [49, 50, 36] – exist but they are often cumbersome to execute and are classifier-specific. When an MPA relies on a third-party mobility prediction engine, it typically does not know which predictors are used or whether they are calibrated or not. Lastly, using a single binary classifier might also be prone to errors. It is indeed known in the literature that ensemble classifiers – i.e., combination of classifiers that perform the same task – provide for better performance [46, 12, 16, 17]. In this paper, we rely on a specific form on ensemble learning known as *bucket of models* [1, 14]. Thereby, the most suitable classifier for a given task is selected among a set of candidates. More specifically, we provide the following contributions.

- We describe the design and implementation of an ensemble classifier – called *LOTUS*[1] – that estimates the level of trust of a next-place prediction. LOTUS takes the current prediction and its class membership probability – along with the current place – as input from a mobility prediction engine. It then outputs a binary decision that indicates whether the current mobility prediction is correct or not.

- We evaluate the performance of LOTUS and show that it can achieve better performance than binary classifiers that rely on class membership probability only. We ground our analysis on two large, publicly available data sets of human mobility records: The Nokia data set [29] and the Device Analyzer data set [45]. To the best of our knowledge, this is the first work that systematically analyzes the uncertainty of next-place predictions on these two data sets.

- We conduct a questionnaire-based user study that (1) affirms the general value users see in the concept of MPAs; (2) confirms our understanding that different usage scenarios will require significantly different levels of trust for a next-place predictor; and (3) reveals that the required level of trust depends on the potential for negative consequences stemming from an MPA's actions.

The remainder of the paper is structured as follows. We first review related work and outline the novelty of our approach. We then present our level of trust estimator and discuss its performance. Further, we describe our questionnaire study by highlighting the hypotheses we aim to verify, the study design, and the obtained results. Although our questionnaire study is part of the motivation of this work, we discuss it after presenting LOTUS for the sake of clarity. Finally, we discuss the current limitations of our approach and outline possible directions for future work.

2. RELATED WORK

Our work lies at the intersection of different research areas, in particular: human mobility prediction, predictability and uncertainty of human behavior, and calibration of machine learning classifiers.

A large number of approaches to solve human mobility prediction problems have been proposed in the literature. For instance, several authors have focused on the next-place prediction problem [39, 2, 5, 12, 33], which is also considered in this work as an example. Other authors used social-ties to improve the predictive power of predictors or to extract users' behavior [11, 28]. Predicting household occupancy is related to the next-place prediction task – whereas only two places (*home* or *not home*) are of relevance. Several solutions to detect and predict household occupancy exist [40, 25, 23]. Furthermore, predicting which applications on users' mobile device will be used next has also gained attention in recent years [47, 12, 41, 48]. The techniques described above are all complementary to our approach. They focus on predicting human activities in different application scenarios but do not consider or incorporate information about predictions' uncertainty. LOTUS extends the capability of these predictors by determining how much trust should be given to each prediction.

Some approaches focusing on predicting human behavior, activities, or mobility also incorporate uncertainty information. For instance, in his seminal work, Horvitz [19] proposed *LookOut*, a system for scheduling and meeting management. Whenever the user receives an e-mail, LookOut assists her with scheduling an appointment suggested in the e-mail. LookOut assigns class membership probabilities of user intentions by leveraging a linear SVM classifier. A simple, threshold-based binary classifier is then used to decide whether an action should be executed or not. A similar approach has been used by other authors too [40, 12, 41]. We compare LOTUS to the threshold-based approaches used in previous work and demonstrate that LOTUS can achieve better performance.

Besides the class membership probability, LOTUS uses the *Instantaneous Entropy (IE)* of a user as an additional indicator of the level of trust of a prediction. IE has been proposed as a metric to identify situations in users' daily life that exhibit low predictability [32]. Thereby, McInerney et al. [32] have built upon previous work on the predictability of human mobility [42, 20, 30, 11]. Whereas this previous work focused on determining how predictable a user is in general, McInerney et al.'s goal is to characterize the predictability of a user at each time instant. We leverage the information captured through the IE metric by including it as one of the features in LOTUS. We further consider a level-of-trust estimator that relies on IE only and compare its performance to that of LOTUS and other classifiers.

Besides the work more closely related to the prediction of human mobility and behavior in general, there is a large body of literature dealing with the uncertainty of the output of a classifier in the machine learning domain. Many of the classifiers commonly used in the literature indicate the class membership probabilities [49] of each classification. However, some classifiers do not provide these probabilities [37] and others estimate them poorly [7, 49, 13]. A well-known classifier that does not provide class membership probabilities "natively" is SVM. Platt has however presented a method – called *Platt Scaling* – that allows a SVM classifier to compute such probabilities [37]. Our approach is complementary to Platt Scaling and similar techniques because LOTUS is agnostic to the specific algorithms used by the mobility prediction engine. Zadrozny and Elkan further address the issue of poor class membership probability estimates provided by decision trees and Bayesian classifiers [49]. The authors compare ten different techniques for estimating class membership probabilities. The main outcome of the authors' study is that major changes to the probabilities given by a standard decision tree and naïve Bayesian classier are required to accurately capture class membership probabilities. They thus propose a method that uses *isotonic regression* to estimate class membership probabilities for a multi-class problem [50]. To do so, the technique proposed by the authors transforms a multi-class classification problem into several binary classifications. This is done instead of attempting to derive probability for each class directly.

[1] Level Of TrUst eStimator

The authors calibrate the class membership probabilities for each binary classification problem separately and then assemble them to obtain multi-class probabilities. In contrast to that, in our work we do not require to break down the multi-class classification problem of which place will be visited next. Our approach is agnostic to the specific predictors used by the prediction engine and does not need to know a priori the set of output classes.

Following the same line of Zadrozny and Elkan's work, Niculescu-Mizil and Caruana examine ten supervised machine learning classifiers and their ability to estimate class membership probabilities [36]. The authors show that neural networks and bagged trees are already able to estimate these probabilities accurately without any additional calibration. Classifiers such as SVM and random trees, however, do require an additional calibration after which both classifiers achieve better estimations of the class membership probabilities. The authors perform calibration by utilizing the two methods Platt Scaling [37] and isotonic regression [50]. Their evaluation shows that the former one requires less data for calibration while the latter one achieves better calibration results if a substantial amount of data is available. We consider these and other classifier calibration approaches to be complementary to our work. Classifier calibration helps improving the reliability of the class membership probabilities of machine learning classifiers. We use these probabilities as input to LOTUS, so their improvement is likely to also improve LOTUS's performance. In our current work, we use class membership probabilities from uncalibrated classifiers, as also done in previous work on mobility prediction. Investigating LOTUS's performance when calibrated class membership probabilities are available is part of our future work.

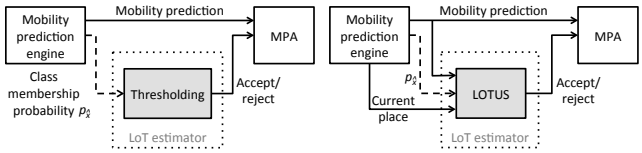
3. LOTUS: ESTIMATING THE LEVEL OF TRUST OF NEXT-PLACE PREDICTIONS USING ENSEMBLE LEARNING

As outlined above, several existing classifiers provide class membership probabilities that quantify what we refer to as the *level of trust* of a prediction. For instance, the implementation of the SVM classifier in the widely used *scikit-learn* Python library provides, for each classified sample, the corresponding class membership probability.[2] Authors working in the human mobility prediction domain have also investigated ways to estimate how "predictable" users are in general or at certain time instants [42, 20, 30, 32].

In this work, our goal is to design a *level of trust estimator* that determines whether the current output of a next-place predictor should be trusted (accepted) or not (rejected). We envision this component to be able to improve the behavior of MPAs and we place it between the mobility prediction engine and the MPA itself, as exemplarily shown in Figure 1b. We focus on the next-place prediction problem because it is one of the most commonly addressed problems in studies of human mobility [2, 26, 33, 39]. Our approach, however, can easily be extended to address other human mobility prediction problems.

Our approach, dubbed LOTUS, relies on ensemble learning. In particular, we leverage the concept of a bucket of models, which is a special form of ensemble learning [1, 14]. Ensemble learning algorithms are known to have a better generalization ability than single predictors [46]. We thus expect LOTUS to obtain better performance than approaches used in previous work and that rely on simple thresholding [19, 12].

In the remainder of this section, we describe the design of LOTUS and the novel *score function* that is uses to determine which



(a) Thresholding. (b) LOTUS.

Figure 1: Sketch showing a level of trust estimator placed between the mobility prediction engine and the MPA.

classifier to use for a specific user and application scenario.

3.1 LOTUS: Design and Implementation

LOTUS trains and evaluates a set of classifiers (bucket of models) and selects among them the most suitable one to be used for a given user and application scenario. We configure LOTUS by including the following six state-of-the-art machine learning predictors in the set of classifiers: Support Vector Machine (SVM) [8, 46], k-Nearest Neighbor (k-NN) [8, 46], Classification and Regression Trees (CART) [9, 46], Naive Bayes (NV) [46], Linear Regression (LR), and Gradient Boost (GB). We choose these approaches because they are representative of different classification strategies. Also, four of these six predictors – SVM, k-NN, CART, and NV – are listed in the top-10 list of algorithms for data mining reported by Wu et al. [46]. We rely on the implementations of these classifiers available from the *scikit-learn* library.[3]

The input to the classifiers running within LOTUS is a set of N_f features $\mathcal{F} = \{f_1, f_2, \ldots, f_{N_f}\}$. The first two features are the current output of the mobility prediction engine. In particular, f_1 is the prediction itself, which we indicate with $\widehat{x}$, while f_2 is the estimate of the corresponding class membership probability, which we indicate as $p_{\widehat{x}}$. Please note that although the values of $\widehat{x}$ and $p_{\widehat{x}}$ change in principle at every time step t_i, we omit the subscript i, for simplicity. The next three features are: the current place of the user (`f_cur_place`); information about the current day being a weekday or a day of the weekend (`f_weekday`); and the number of places already visited on the current day (`f_places_today`). Recent studies have shown that using these three features (in combination with different classifiers) provides for the highest average performance in predicting the next place across a large number of user [6]. Finally, the sixth feature is the IE value computed as specified by McInerney et al. [32]. The IE metric is computed over the sequence of previously visited places (including the current one).

The output of LOTUS is a binary decision about whether the current next-place prediction is correct or not. To determine which classifier to use at runtime to make this decision, LOTUS first trains the set of candidate classifiers mentioned above on a subset of the input data. A further, non-overlapping subset of the input data is then used to evaluate the performance of each classifier. Users (or their MPA) can specify the minimum level of trust that predictions must have to be used to trigger MPA's actions. LOTUS selects, among the set of candidates, the one that provides the best performance given the users' requirements.

The rationale behind this approach is that classifiers' performance might vary depending on the mobility patterns of the user and the particular application scenario, i.e., how much tolerance the user has for the MPA to either fail to perform an action or perform a wrong one. The best performing classifier is hence the one that maximizes the score function that we introduce in Section 3.2. This

[2] `http://scikit-learn.org/stable/modules/svm.html`

[3] `http://scikit-learn.org/stable/`

score function tunes the precision achieved by the classifiers according to the preferences of the user. In Section 6, we address the problem of how to quantify users' preferences in terms of level of trust and depending on the application scenario.

Since the best performing classifier may change over time, a periodical repetition of the training and evaluation phase of the individual classifiers is necessary. In the context of this work, we assume that LOTUS selects the best performing classifier after an initial training and evaluation phase and subsequently uses it at runtime. Determining appropriate retraining intervals is part of future work.

To train and evaluate the classifiers, LOTUS assumes information about the current place of the user to be available – both during training and at runtime. This information can be extracted applying standard algorithms to data that can be collected on common mobile devices (e.g., Wi-Fi scans or GPS coordinates) [4, 21, 34]. Furthermore, the current place of the user is a feature often used by next-place predictors as part of their input data [5, 39, 35, 33, 12]. We thus assume for simplicity that the mobility prediction engine provides this information – as illustrated in Figure 1b – although it can also be obtained by other components.[4]

3.2 LOTUS: Score Function

As explained above, LOTUS selects the best performing classifier out of a set of candidates. These classifiers estimate whether the current prediction is correct or not, i.e., they classify the input as belonging to one of two classes, the *correct* and the *incorrect* classes.

In the training phase, the classifiers take as input both the feature set $\mathcal{F}$ and the ground-truth data (i.e., the current place). This way, they can compute their internal parameters so as to optimize their classification accuracy, which is defined as the ratio of the number of correct classifications and the total number of classifications. In the evaluation phase, the classifiers take as input only the feature set $\mathcal{F}$ – which is computed over a different set of input data than the one used for training. Ground-truth data is used in the evaluation phase only to assess the actual performance of the classifiers.

We use three metrics to describe this performance. The first metric is the precision of the classifier, indicated as PRE. It is computed as the ratio of the number of correct predictions that are also classified to be correct (i.e., the so-called true positives, TP), and the total number of predictions that are classified to be correct, which includes also the false positives (FP) [8]. Thus:

$$PRE = \frac{TP}{TP + FP}. \tag{1}$$

The second metric, which is known as Negative Predictive Value (NPV), is the ratio of the number of correct predictions that are classified to be incorrect (i.e., the so-called true negatives TN), and the total number of predictions that are classified to be incorrect, which also includes the false negatives (FN). Thus:

$$NPV = \frac{TN}{TN + FN}. \tag{2}$$

At runtime, the values PRE and NPV, derived from our best performing classifier and feature set, represent the calibrated class membership probabilities for predictions that are classified as correct and incorrect, respectively.

The third metric, which we refer to as Positive Prediction Rate (PPR), is the ratio between the number of predictions that are classified as correct, irrespective of whether the prediction is actually correct or not, and the total number of predictions. The numerator of this ratio is thus the sum of true positive (TP) and false positive (FP) classifications while the denominator is the total number of classifications, i.e. the sum of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) classifications. Thus:

$$PPR = \frac{TP + FP}{TP + TN + FP + FN}. \tag{3}$$

A false positive classification occurs when LOTUS classifies a prediction of the mobility prediction engine to be correct when it is actually incorrect. Thus, the metric PPR expresses the total number of instances for which LOTUS states that a prediction of the mobility prediction engine is correct. Since we assume that an MPA may execute an action when LOTUS estimates a prediction to be correct, PPR actually represents the upper bound for the total number of instances in which such an action is executed by the MPA, regardless of whether executing this action is correct or not. Without a level of trust estimator the MPA would assume all predictions to be correct, and thus potentially execute autonomous actions whenever the mobility prediction engine says so. This is equivalent to the case in which a *dummy* classifier that estimates all predictions to be correct is used, resulting in $PPR = 1$ (100%).

As we saw in our survey, described in Section 6, certain use cases (e.g., file prefetching) might see a user preferring to have the MPA perform actions whenever possible, regardless of whether the action is actually correct or not ($PPR = 1$). For other use cases (e.g., home automation), users may instead prefer to have the MPA perform an action only when the level of trust in the prediction is very high (highest achievable PRE).

To capture differences in the level of trust requirements given by application scenarios, LOTUS chooses the classifier to use at runtime according to the score function that combines the two performance metrics PPR and PRE. We indicate this score function with the symbol SF_w and define it as:

$$SF_w = PRE + tan(w \times 45) \times PPR, \tag{4}$$

where *tan* represents the trigonometric function tangent and the weight w is a value between 0 and 1 that captures the preferences of the user. LOTUS selects the classifier with the highest value of SF_w to be the classifier to use at runtime.

The rationale behind the definition of LOTUS's score function is the following. When the weight w is set to 0, the value of SF_w is equal to PRE. Thus, the classifier that provides the highest precision (i.e., the highest true positive ratio) is chosen and used at runtime. This implies that the number of false positives "seen" by the MPA is decreased with respect to the case in which LOTUS is not used. Since a false positive classification triggers the unnecessary execution of an action, the total number of these actions is reduced. This clearly comes "at the cost" of an overall lower number of actions executed by the MPA (lower PPR). The weight w should thus be set to 0 when the user prefers the MPA to execute autonomous actions only when it is very confident that the action is really necessary (i.e., that the prediction of the mobility prediction engine is correct). On the other hand, the weight w should be set to 1 when the user prefers the MPA to execute autonomous actions irrespective of how much trust is given to a particular mobility prediction. While in principle any value between 0 and 1 can be chosen for w, we consider in the following and in our evaluation values of w equal to 0, $1/6$, $2/6$, $3/6$, $4/6$, $5/6$, and 1. This corresponds to a 7-point Likert scale used in our questionnaire, which is described in Section 6. We exploit the data collected through our questionnaire to derive realistic values for w.

We illustrate this point for a specific example, reported in Fig-

[4] In this case, however, it must be ensured that both LOTUS and the mobility prediction engine use the same set of relevant places.

ure 2. The figure shows the case in which LOTUS is used to classify the predictions of a mobility predicting engine that achieves a prediction accuracy of 40%. The x-axis reports the value of PPR obtained by LOTUS's classifiers while the y-axis reports the corresponding precision PRE. The point Q ($PPR = 40\%$, $PRE = 100\%$) indicates the performance that would be achieved by a perfect classifier. This classifier would indeed identify all the correct predictions output by the mobility prediction engine, which are 40% of the total, and thus have a PPR of 40% (where FP=0) and a PRE value of 100%. The point P ($PPR = 100\%$, $PRE = 40\%$) in the figure reports the performance that the dummy classifier would achieve, which also corresponds to the case in which LOTUS is not used. Indeed, if 40% of the mobility predictions are correct and the classifier estimates all predictions to be correct ($PPR = 100\%$), then its precision is necessarily equal to 40%.

For a given value of PPR, the corresponding point on the curve labeled as "Upper boundary" in Figure 2 indicates the highest precision achievable by any classifier. Moving beyond the upper boundary requires improvements of the mobility prediction engine's performance. The colored region in Figure 2 is the region in which the performance of the classifier chosen by LOTUS should lie. We define a "Lower boundary" as the boundary that is achieved by a random classifier.

Let us assume in this example that w is equal to $3/6$, i.e., the solid line labeled with $w = 3/6$ represents the "Decision boundary". Thus, the entire colored region above the decision boundary can be used to search for a suitable classifier. A suitable classifier is in this case any classifier that can improve (+X%) upon the 40% precision of the dummy classifier. The costs for this improvement in terms of the reduced PPR, i.e., number of the executed actions, must be lower than $\frac{X}{tan(3/6\times45)}$. In this case the term $tan(3/6 \times 45)$, which is equal to 0.4142, describes the slope of the decision boundary. In other words, for an improvement of PRE of 0.41% the user is willing to give up no more than 1% of the executed actions (PPR). The value 45 describes the degree of the angle (slope = 1) between the decision boundary for $w = 0$ and $w = 1$. The rationale behind setting $w = 1$ to the angle of 45 degree is that no classifier can reach the area above the upper boundary. Therefore, we define the decision boundary for $w = 1$ as the line that connects the points P and Q. As we observe, our function SF_w then divides this area into seven parts (equal angles). In future, we plan to develop a non-linear function that reflects users' preferences more accurately than a linear function by conducting further studies.

Figure 2 exemplarily shows the performance of three fictive classifiers – indicated as *A*, *B*, and *C*. Since w is equal to 0.5, the classifier B cannot be chosen, because its achieved SF_w value is lower than that of the dummy classifier. Both A and C are above the decision boundary and are preferred over the dummy classifier. In this example, we observe that the classifier A is further from the decision boundary than classifier C ($d_A > d_C$). This is reflected in a higher value of the function $SF_{3/6}$. Therefore, the best solution for the given user's preferences is classifier A.

4. EVALUATION SETUP

We now describe experimental setup we choose for demonstrating the performance of LOTUS. The corresponding evaluation results are then presented in the next section.

4.1 Data Sets

We run our evaluation on two large, publicly available data sets to which we refer as the *Nokia data set* and the *Device Analyzer data set*. The Nokia data set has been collected during the Nokia

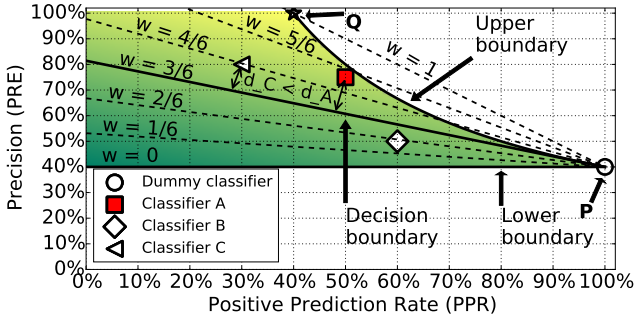


Figure 2: An illustrative example of how the score function allows LOTUS to select the best performing classifier. The example refers to a situation in which the mobility prediction engine achieves a precision of 40% and the user-defined weight w is set to 0.5 (3/6).

Lausanne Data Collection Campaign (LDCC) [22]. Part of this data set was made available to the public in 2012 in the context of the Nokia Mobile Data Challenge (MDC) [29]. The entire data set was released to the public in 2013 and is now managed by the IDIAP Research Institute (**`http://www.idiap.ch/`**). The Nokia data set contains data collected from the mobile phones of 141 users for a period of about 18 months. The collected data includes, for instance, GPS traces, records of Wi-Fi scans, Bluetooth scans, phone calls, and user demographics including age and gender. The most commonly visited places for each user are also available. According to the documentation accompanying the data set, these were determined using an approach similar to [34], and were validated by the users themselves.

The Device Analyzer data set [45] has been first made publicly available in the context of the UbiComp/ISWC 2014 Programming Competition (**`http://goo.gl/6w3GwY`**). The publicly available version of the data set contains sensor data from over 17,000 users across the world. We have used a tool available along with the data set to select a subset of 790 users for whom at least 10 month of sensor data is available. We then applied a heuristic for extracting users' relevant places similar to the one used by Bao et al. [3]. Finally, we consider 268 users with at least 100 relevant places visits and at least 10 unique places recorded.

To run our evaluation, we split the data of each user in four subsets of equal size. We call these subsets T_1, T_2, T_3, and T_4. We use T_1 to train the mobility predictor. We then use the trained predictor to compute NP predictions using T_2 and T_3 as input. The output of the mobility predictor is then given as input to LOTUS. Furthermore, it uses data from T_2 (T_3) and the corresponding output of the mobility predictor to train (evaluate) its set of classifiers. Finally, we use data from T_4 to evaluate the performance of the level of trust estimators considered in this work.

4.2 Mobility Predictor

We evaluate LOTUS's ability to discriminate correct and incorrect estimates in the context of the next-place prediction problem, i.e., the problem of predicting the next place that will be visited by a user. For evaluation purposes, we assume the mobility prediction engine uses SVM as the mobility predictor that computes next-place predictions. We assume this predictor uses three input features: (1) the current place of the user (`f_cur_place`), (2) information about the current day being a weekday or a day of the weekend (`f_weekday`), and (3) the number of places already visited on the current day (`f_places_today`). We use SVM in combination with this set of features because recent studies have

shown that this configuration provides for the highest average performance across a large number of users [6].

4.3 LOTUS's competitors

We compare LOTUS to five other techniques to estimate the level of trust of mobility predictions. These are: (1) *majority vote* (0-R); (2) thresholding over the class membership probabilities (T_{TH}); (3) thresholding over the IE metric (IE_{TH}); (4) a dummy estimator that assumes all mobility predictions to be correct; and (5) a modification of LOTUS to which we refer to as the *Best Accuracy* (BA) estimator and that is described below.

The majority vote (0-R) estimator observes the accuracy achieved by the mobility prediction engine at runtime. If the accuracy is $> 50\%$, then the 0-R estimator assumes the mobility predictor is always correct and always classifies the prediction of mobility predictor as correct (and vice versa if the accuracy is $\leq 50\%$). The T_{TH} approach estimates the output of the mobility predictor to be correct if the class membership probability corresponding to the current prediction, indicated as $p_{\hat{x}}$, is higher than a pre-specified threshold TH [19, 40, 12, 41]. The IE_{TH} approach is similar to T_{TH}, but applies a thresholding over the computed value of IE [32] instead of over $p_{\hat{x}}$. For both T_{TH} and IE_{TH} we experiment with different thresholds: 20%, 40%, 60%, and 80%. The dummy estimator simply assumes that all mobility predictions are correct. This corresponds to the case in which no LoT estimator is used. Lastly, BA is a modification of LOTUS in which the best performing classifier is selected as the one that achieves the highest classification accuracy.

5. EVALUATION RESULTS

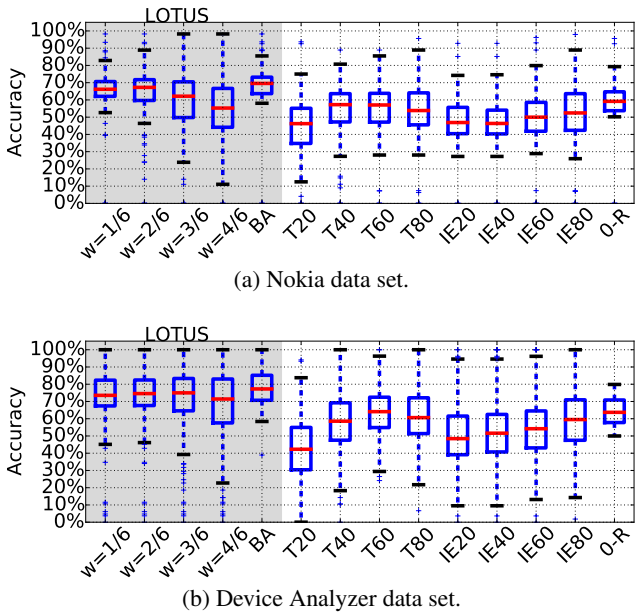
We discuss now the performance achieved by LOTUS and the other level of trust estimators introduced in the previous section. First, we show that LOTUS can adapt to different users' mobility patterns and available input data by choosing different classifiers to operate at runtime. Second, we compare classification accuracy of all estimators. Third, we evaluate for how many users in both data sets each of the estimators achieve a better performance than the dummy classifier.

5.1 Classifiers chosen by LOTUS

LOTUS is capable of selecting a classifier depending on the specific mobility pattern of a user as well as depending on the application scenario. We show exemplary that this is actually the case for the Nokia data set. To this end, we compute the percentage of cases in which each of the classifiers included in LOTUS's set of classifiers has been selected as the best one. The computation is performed for all users and weights w. We inspect the results (omitted due to space constraints) and observe that in about half of the cases either k-NN or LR is chosen (26% and 25%, respectively). In 17% of the cases CART is selected. In the remaining cases the choice is almost equally split among NV (12%), GB (11%), and SVM (10%). This data shows that none of the six considered classifiers can be considered to clearly be the "best" one for all users and weights. This supports our design choice of making LOTUS dynamically select the best classifier out of a set of candidates.

5.2 Classification Accuracy

Figure 3 shows the classification accuracy achieved by the considered level of trust estimators. Results are reported for both the Nokia data set (a) and the Device analyzer data set (b). The boxes cover the 25th and 75th and the whiskers the 5th and 95th percentiles. The markers within the boxes correspond to median values. We report the results for the weights w: $1/6$, $2/6$, $3/6$, and $4/6$. This corresponds to the weights identified as relevant in our questionnaire-based study (cf. Section 6).



(a) Nokia data set.

(b) Device Analyzer data set.

Figure 3: Classification accuracy achieved by the considered LoT estimators.

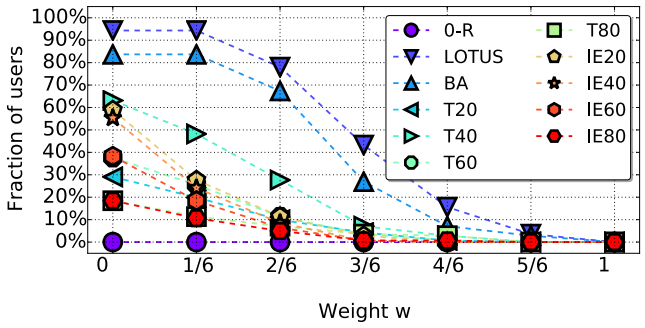
As expected, we observe that LOTUS achieves higher accuracy than other approaches. Thus, LOTUS is more effective than its competitors in identifying which predictions are correct and which are not. For instance, for the Nokia data set the highest median accuracy is achieved by LOTUS with $w = 2/6$ and BA (67% and 69%, respectively). The next best median accuracy (excluding LOTUS) is achieved by the $0-R$ estimator (59%). Similarly, for the Device Analyzer data set the highest median accuracy is achieved by LOTUS with $w = 3/6$ and BA (75% and 78%, respectively). The next best median accuracy (excluding LOTUS) is achieved by the $T60$ and $0-R$ estimators (64%). LOTUS's superior performance is due to the fact that it relies on ensemble learning.

From Figure 3 we can further observe that for both the Nokia data set and the Device Analyzer data set LOTUS's median accuracy decreases as the weight w increases. This decrease in performance is due to the fact that a higher weight w makes LOTUS classify more predictions as correct. In other words, LOTUS will risk to classify more predictions as correct (even if they are incorrect) to avoid the risk of erroneously classifying correct predictions as incorrect. This in turn corresponds to a situation in which a user is more willing to accept the MPA to trigger actions unnecessarily than to accept the MPA not to trigger an action when it is supposed to do so.

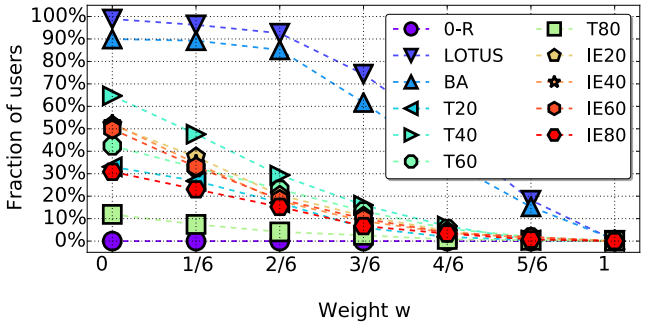
Lastly, Figure 3 also shows that the classifiers tend to perform better over the Device Analyzer data set rather than the Nokia data set. We believe this is mainly due to the fact that the average number of relevant places for users in the Nokia data set is substantially higher than for users in the Device Analyzer data set. This implies that classifiers operating on the Device Analyzer data set must deal with a substantially lower number of output classes.

5.3 Benefits from Using Level of Trust Estimators

In this work, we assume that an MPA triggers an action only if a level of trust estimator classifies the current next-place prediction

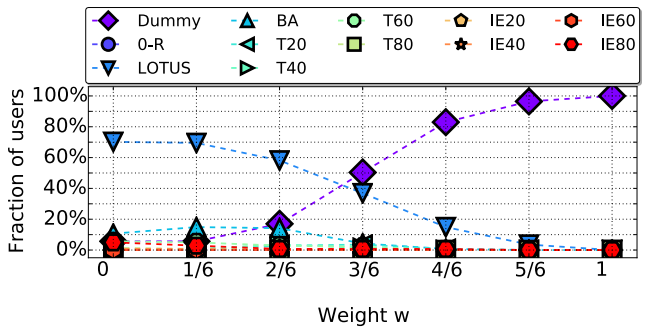


(a) Nokia data set.

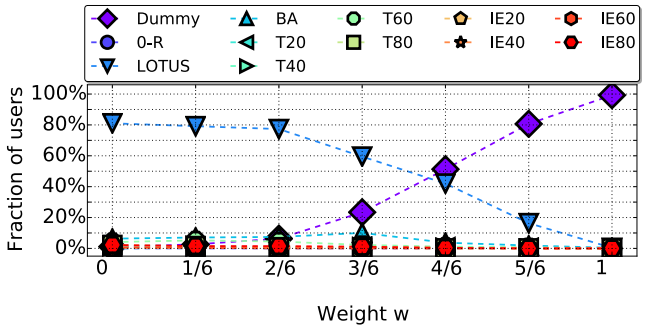


(b) Device Analyzer data set.

Figure 4: Fraction of users for whom a level of trust estimator achieves a better performance than the dummy classifier in terms of the achieved score and with respect to weight w.



(a) Nokia data set.



(b) Device Analyzer data set.

Figure 5: Comparison of the different estimators in terms of the achieved score. For each estimator the figure shows the fraction of users for those the corresponding estimator achieves the best score with respect to weight w.

as correct. The ratio of these positive classifications over the total number of classifications performed by an estimator is given by PPR. The level of trust of each positive classification is given by the precision PRE of the classifier. Furthermore, the level of trust of each negative classification is given by the negative predictive value NPV of the classifier. If no level of trust estimator is used – or, equivalently, the dummy estimator is used – PPR is equal to 100% and PRE corresponds to the accuracy of the mobility predictor.

Figure 4 shows the gain achievable from using LOTUS or any other estimator with respect to the case in which the dummy estimator is used. Each curve in Figure 4 corresponds to the performance of an estimator. The performance of each estimator is computed as the value of SF_w defined in Equation 4. Each point on a curve represents the percentage of users for whom the corresponding estimator provides, for the weight indicated on the x-axis, a higher value of SF_w than a dummy classifier.

From Figure 4 we can observe that LOTUS outperforms, for both the Nokia data set and the Device Analyzer data set, any other estimator for all weights but $w = 1$. This is because, as expected, LOTUS maximizes the value of SF_w for the highest number of users. As already mentioned on previous occasions, the case $w = 1$ corresponds to a situation in which users prefer MPAs to execute all potentially useful autonomous actions irrespectively of whether or not the mobility prediction is correct. Thus, in this case there is in any case no need to use any level of trust estimator.

Each curve in Figure 5 shows the percentage of users for whom the corresponding classifier achieves the highest value of SF_w among all considered estimators. For instance, for $w = 2/6$ LOTUS achieves the highest value of SF_w for 57% and 76% of users for the Nokia and Device Analyzer data set, respectively. As the weight w increases, the percentage of users for whom using the dummy classifier represents the best option also increases.

6. QUESTIONNAIRE-BASED USER STUDY

The motivation of our work is based on four hypotheses that we formulate as follows:

- *H1 (Utility): Users see utility in having MPAs take autonomous actions on their behalf.*
- *H2 (Context-dependence): A user-acceptable error rate for an MPA depends on the underlying use case scenario.*
- *H3 (Confidence): Users want MPAs to estimate the level of trust of their predictions when taking autonomous actions.*
- *H4 (Adequacy): The level of trust needed depends on the potential negative consequences from autonomous actions.*

Although some of these hypotheses might be obvious, we provide first quantitative evidence by conducting a questionnaire-based user study with 188 participants from 18 countries across six continents. To explore H2 and H4, we selected three representative scenarios in our study: *home automation*, *providing traffic updates*, and *prefetching mobile application data*. These three scenarios not only span a broad spectrum of use cases but also feature very different risks, i.e., the negative consequences from the unnecessary execution of tasks are very different in each scenario. Participants received short explanations similar to the sections below, describing the three scenarios.

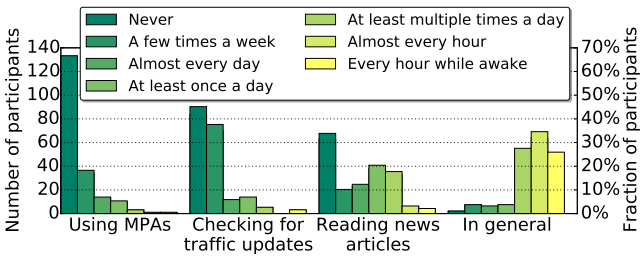


Figure 6: Mobile device usage statistics indicated by the participants of the study.

Home Automation.

In the context of home automation, the MPA on a user's mobile device predicts that the user will return home soon. Using this prediction, the MPA remotely switches on the heating so that the user returns to a warm home. Similarly, the MPA automatically switches off the heating when it detects that the user is about to leave the house soon. Incorrect predictions may lead to a waste of energy – and hence money – if the heating system is turned on too early (or off too late), or a loss of comfort when it is turned on too late (or off too early).

Providing Traffic Updates.

In order to provide traffic updates, the MPA on a user's mobile device predicts that the user will soon go to another place. The MPA automatically verifies current traffic conditions en-route to this new place and alerts the user when it is time to leave. Incorrect predictions may lead to unnecessary notifications for wrongly predicted trips (which might annoy the user) or a lack of alerting the user to leave, in case the MPA misses a trip (which may result in the user being late).

Prefetching Mobile Application Data.

Given Wi-Fi connectivity at the current place, the MPA will start prefetching data for those applications that the user will most likely use at the next predicted place, or while en-route. Incorrect predictions in this scenario may lead to the prefetching of unneeded data (which may lower battery life), or to the lack of availability of needed data (or additional costs from using cellular data instead).

6.1 Methodology

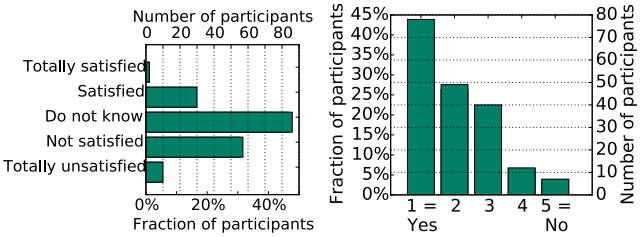
The questionnaire is divided into four parts: Part 1 asks for demographic data (e.g., age, gender, country of origin, country of residence); Part 2 collects information about the participant's mobile device usage behavior (e.g., how often they use a specific feature on their mobile device); Part 3 focuses on *if* and *when* a participant would like to have an autonomously acting MPA (H1 and H2); and Part 4 focuses on issues involving the uncertainty of an MPA's decisions (H3 and H4). An anonymized version of our questionnaire is available at **`https://goo.gl/1c8UHM`**. After testing several iterations of the questionnaire design with volunteers not involved in the design process, we distributed the questionnaire both by word-of-mouth through our personal network, as well as by posting links to it on social media and mailing lists.

6.2 Results

We received a total of 188 completed replies to our questionnaire. Table 1 summarizes the demographics. Due to our recruitment channels, the large majority of our participants are male (79%) and are members of the academic community (90%). More than half are heavy phone users who use their mobile device almost every hour (see Figure 6). We also asked participants how often they currently use an MPA, e.g., Google Now, and how satisfied they are with it. The results reported in Figure 6 and Figure 7a indicate that the majority of participants never use an MPA, and those who have done so are largely not satisfied with them.

Table 1: Demographic groups and their statistics.

Label	Description	Total	Fraction
`u_total`	Number of participants	188	100%
`u_res_c`	Number of residence countries	18	100%
`u_origin_c`	Number of origin countries	40	100%
`g_female`	Female	38	20%
`g_male`	Male	148	79%
`p_student`	Student	50	27%
`p_phd_student`	Research assistant or PhD student	83	44%
`p_postdoc`	Postdoctoral researcher	17	9%
`p_faculty`	Faculty member	19	10%
`p_employee`	Employee	16	9%
`p_other`	Other	2	1%
`a_22`	Below 22 years old	2	1.1%
`a_22_27`	Between 22 and 27 years old	70	37.2%
`a_28_33`	Between 28 and 33 years old	71	37.8%
`a_34_38`	Between 34 and 38 years old	24	12.8%
`a_39_44`	Between 39 and 44 years old	8	4.3%
`a_45_50`	Between 45 and 50 years old	2	1.1%
`a_50`	Above 50 years old	5	2.7%



(a) Users' satisfaction in using MPAs.

(b) Users' willingness of living without an MPA.

Figure 7: Satisfaction in using an MPA and willingness of living without it.

6.2.1 Perceived Value of Autonomous Actions

While most participants were happy to live without an MPA (see Figure 7b), participants generally found the concept of an MPA to be helpful (see Figure 8), confirming our hypothesis "H1: Utility". Only 7% indicated that having an MPA that autonomously executes tasks on their behalf would not help them at all. For the three scenarios, the perceived value of an MPA was even higher (see Figure 8), in particular for the scenario of prefetching data.

To investigate hypothesis "H2: Context-dependence", we asked participants to indicate what percentage of MPA errors they would tolerate when it is autonomously executing a specific task before they stopped using the MPA. Figure 9 reveals that for the home automation scenario, almost two-thirds of participants would only accept an error rate of less than 10%. In the case of traffic updates, it was 50% of participants. For data prefetching, participants were willing to accept substantially more errors than for the other two application scenarios. This seems to confirm H2.

6.2.2 Trustworthiness

To verify hypotheses "H3: Confidence" and "H4: Adequacy",

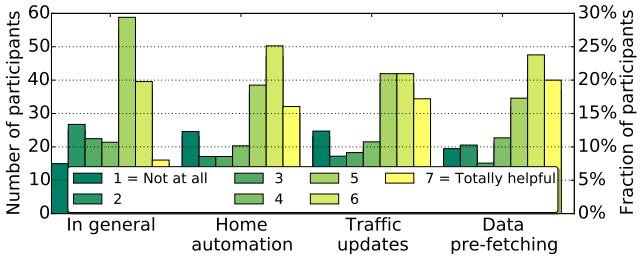


Figure 8: The users' perceived value of having support from an MPA for different application scenarios.

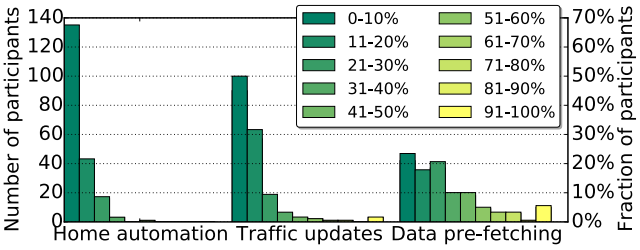


Figure 9: Percentage of acceptable errors according to users' opinion for the different application scenarios.

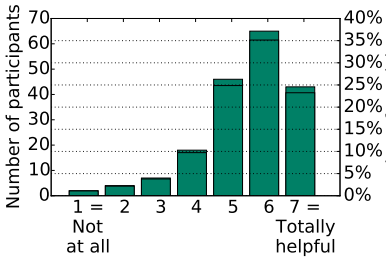


Figure 10: The users' perceived value of having an MPA that is capable of estimating the level of trust of a prediction.

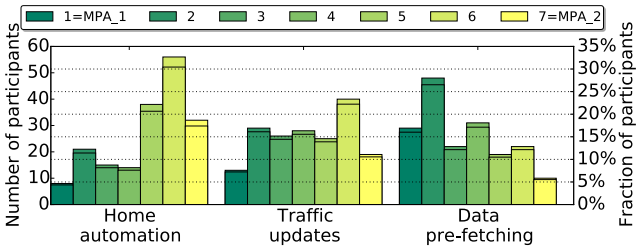


Figure 11: Users' preferences for a particular behavior of an MPA for the three considered application scenarios.

we asked participants to assume that for each of the three application scenarios, there were two MPAs on the market: MPA_1 does not use any kind of level of trust estimation and hence autonomously executes tasks whenever it predicts a change of place. MPA_2 instead uses a level of trust estimator and only executes a task if the level of trust in its prediction is very high. Participants were asked to indicate their preference between these two MPAs on a 7-point Likert scale.

In different scenarios, participants preferred different MPAs (see Figure 11). In the case of home automation, and to a lesser extent for traffic updates, MPA_2 was preferred. For the prefetching scenario, participants preferred MPA_1. This seems to confirm H4. Almost 60% of participants found the idea of an MPA that can estimate the level of trust of its predictions valuable or very valuable (see Figure 10), confirming H3.

6.3 Discussion

Over 90% of our study participants were owners of either an Android, Apple, or Microsoft smartphone. While most of these will most likely run a recent OS-version that comes with an MPA (e.g., Google Now or Microsoft Cortana), almost 70% of all participants did not use an MPA at all. Almost 45% of participants indicated that they were willing to live without an MPA. However, 93% of our participants saw the benefit in having an MPA that acts on their behalf. Clearly, there is much room for improvement when it comes to "anticipatory computing" in today's MPAs.

Figure 9 and Figure 11 reveal that the acceptable error rate for incorrectly executed tasks on a user's behalf depends on the underlying scenario. In our data, we also saw a high variability between users, with some being more risk averse than others across all three scenarios. It is hence important to have an MPA that is able to adapt to the required level of trust of a scenario, as well as a user's individual comfort level.

The design of LOTUS incorporates the parameter w for controlling the accept/reject rate of mobility predictions. Our questionnaire-based study provides us with the possibility to derive realistic values for the weight w for the three application scenarios considered in this work. In the following discussion, we primarily focus on LOTUS since it outperforms other Level of Trust (LoT) estimators.

6.3.1 *Home Automation*

The results of the study revealed that almost half of participants indicated their preference between MPA_1 (1) and MPA_2 (7) with a value 6 or higher on a 7-point Likert scale. This corresponds to the weight $w = 1/6$. Nearly two-thirds of participants indicated their preference with (5) or higher, which corresponds to the weight $w = 2/6$. By selecting $w = 2/6$ 82% and 91% of the users will benefit from any of the considered estimators for the Nokia and Device Analyzer data sets, respectively (see Figure 5). Compared to the dummy classifier, which executes all actions, 78% and 91% of the users benefit from LOTUS for the Nokia and Device Analyzer data sets, respectively (see Figure 4). Comparing to other approaches, LOTUS (considering $SF_{2/6}$ only) provides for 59% and 78% of the users the best solution for the Nokia and Device Analyzer data sets, respectively (see Figure 5). We conclude that LOTUS outperforms other level of trust estimators for the majority of users in the home automation scenario. Compared to the dummy estimator, the majority of the users in both data sets benefit from at least one level of trust estimator.

6.3.2 *Traffic Updates*

For the traffic updates scenario, almost 45% of participants indicated their preference with a value 5 or higher on a 7-point Likert scale. At the same time, 60% of them indicated it with a value 4 or higher on a 7-point Likert scale, which corresponds to the weight $w = 3/6$. We focus our discussion on the results for $w = 3/6$ only because the results for $w = 2/6$ are the same as for the home automation scenario. With respect to the dummy classifier, 50% (Nokia) and 77% (Device Analyzer) of users benefit from any other level of trust estimator for the weight $w = 3/6$. Among those individuals, 44% (Nokia) and 75% (Device Analyzer) of them benefit, in particular, from LOTUS. Comparing to other estimators, LOTUS (again considering $SF_{3/6}$ only) provides for 35% (Nokia) and 60% (Device Analyzer) of users the best solution. Given these observa-

tions, we conclude that the majority of users still benefit from at least one estimator in the context of the traffic updates scenario.

6.3.3 Data Prefetching

Lastly, we consider the data prefetching scenario. The results of the questionnaire-based study revealed that 55% of participants indicated their preference with a value 3 or higher on a 7-point Likert scale, which corresponds to the weight $w = 4/6$. With respect to the dummy classifier, only 17% and 48% of users benefit from any of the estimators leveraged in this work for the weight $w = 4/6$ for the Nokia and Device Analyzer data sets, respectively. Comparing to other estimators, LOTUS (again considering $SF_{4/6}$ only) provides for 15% (Nokia) and 41% (Device Analyzer) of the users the best solution. Given these observations, we conclude that the majority of the users will experience best results in terms of the value of SF_w if all actions are executed. However, LOTUS is still able to identify users who benefit from using it in comparison to the dummy classifier.

7. LIMITATIONS AND FUTURE WORK

In the previous sections we have described our level of trust estimator and demonstrated that it is more effective than other approaches previously used in the literature. We now highlight some limitations of this work and outline potential directions for future work.

Comparison LOTUS to More Sophisticated Competitors: LOTUS and the other approaches considered in this study differ in terms of their complexity and technique used to estimate the level of trust. Our results show that LOTUS can outperform its competitors. This is mainly because LOTUS relies on ensemble learning and it can thus outperform single predictors [46]. In our future work, we plan to investigate how LOTUS would perform against more sophisticated competitors. In particular, we plan to design a novel ensemble classifier controlled by an additional rejection classifier trained on (and with runtime access to) the same data as LOTUS. The precision/recall of each of these classifiers would be picked as per domain requirements (similar to picking the weight w). We plan to compare LOTUS with this baseline via perceived user satisfaction in a real deployment (to model the cost of rejection), or at least with respect to raw accuracy/precision/recall performance results. Furthermore, additional data sets – such as KDD'98, which is known to be one of the largest, publicly available data sets containing real-world misclassification costs [18] – can be used to further strengthen our evaluation.

Mobility Prediction Engine as a Black Box: Some of the existing mobility predictors, in particular probabilistic classifiers, provide a possibility to set a class membership probability threshold. This threshold allows the classifier to decide to which class a given prediction should be assigned based on the computed class membership probability [27]. Different settings make the inference more prone to false positives or false negatives. In this work, we consider the mobility prediction engine to be an independent "black-box" component that cannot be modified. Therefore, we compare a set of level of trust estimators under the assumption that they do not have an influence on the parameters and the output of the mobility prediction engine. Removing this assumption, e.g., evaluating the possibility to set the probability threshold of probabilistic classifiers, is a potential direction for our future work.

Considering calibrated Class Membership Probabilities: The level of trust estimators utilized in this work use class membership probabilities provided by the mobility prediction engine. Although these probabilities are known to be inaccurate [7, 49, 13], calibration techniques exist that allow these class membership probabilities to be transformed to represent the true probability of the class. However, not all machine learning classifiers and human mobility predictors support calibration. Several techniques have been proposed to address this shortcoming [49, 50, 37, 36]. Evaluate how LOTUS and its competitors perform with calibrated when class membership probabilities is part of future work.

Subjective Perceived Value of MPAs: The goal of the questionnaire-based study presented in Section 6 was to verify the validity of a set of assumptions we rely on in this work. For instance, that different usage scenarios will require significantly different certainty levels for a next-place predictor and that the required level of trust depends on the potential for negative consequences stemming from an MPA's actions. Although these assumptions seem to partly be intuitive and expectable, our study results provide first quantitative evidence that these assumptions do hold. Furthermore, we obtain quantitative indications on how to set the weight w in different application scenarios. A drawback of our study is that most of the participants have a limited experience with the use of MPAs and the three considered application scenarios. The study results thus reflect subjective opinions of the participants and we consider them a first step towards understanding how practical application scenarios that rely on mobility predictions can benefit from level of trust estimators. Our next step is to implement an MPA that reflects the application scenarios considered in this work and to test it in a real deployment. This field study will allow us to capture individuals' perceived value of using these MPAs in daily life. It further will allow us to quantify the potential costs and benefits for each application scenario caused by an incorrect and correct mobility prediction, respectively.

8. SUMMARY AND CONCLUSIONS

To have users trust an MPA to take actions on their behalf, we need to be able to better control the required confidence of an MPA in its predictions. As we have seen from an online survey involving 188 participants, users have different requirements with respect to the prediction quality of an MPA, depending on different use cases.

In this work, we analyzed whether and to what extent real-world application scenarios benefit from level of trust estimations. In terms of the scenarios, we included (1) home automation, (2) traffic updates, and (3) data prefetching. We designed, implemented, and evaluated LOTUS – an ensemble learning based algorithm for detecting uncertain and probably incorrect mobility predictions. We compared LOTUS to a set of other level of trust estimators on three application scenarios and two large data sets in the context of the next-place prediction task. We showed that LOTUS clearly outperforms its competitors such as thresholding or majority vote. For instance, if individuals' trust requirements are located exactly between the two extrema MPA_1 and MPA_2, then no more than 18% of these individuals will benefit from the best performing competitor estimator. In contrast to that, with LOTUS, at least 43% of individuals benefit. In general, we observe that the higher the accuracy requirements on mobility predictions and thus correctly executed actions are, the more individuals benefit from at least one of the level of trust estimators considered in this work. In particular, for the Nokia data set 82%, 50%, and 17% of the users benefit from at least one of the estimators considered in this work in the context of the home automation, traffic updates, and data prefetching scenarios, respectively.

9. REFERENCES

[1] C. Aggarwal. Outlier Ensembles [Position Paper]. *ACM SIGKDD Explorations Newsletter*, 14(2), 2013.

[2] D. Ashbrook and T. Starner. Using GPS to Learn Significant Locations and Predict Movement Across Multiple Users. *Personal and Ubiquitous Computing*, 7(5), 2003.

[3] X. Bao, N. Z. Gong, B. Hu, Y. Shen, and H. Jin. Connect the Dots by Understanding User Status and Transitions. In *Adjunct Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication (UbiComp Adjunct)*, 2014.

[4] P. Baumann, J. Klaus, and S. Santini. Locator: A Self-adaptive Framework for the Recognition of Relevant Places. In *Adjunct Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication (UbiComp Adjunct)*, 2014.

[5] P. Baumann, W. Kleiminger, and S. Santini. The Influence of Temporal and Spatial Features on the Performance of Next-place Prediction Algorithms. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, 2013.

[6] P. Baumann, C. Köhler, A. K. Dey, and S. Santini. A Population Model for Predicting Human Mobility. In *Adjunct Proceeding of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication (UbiComp Adjunct)*, 2015.

[7] A. Bella, C. Ferri, J. Hernández-Orallo, and M. J. Ramírez-Quintana. Calibration of Machine Learning Models. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, 2010.

[8] C. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[9] L. Breiman, J. Friedman, C. Stone, and R. Olshen. *Classification and Regression Trees*. Wadsworth Publishing Company, 1984.

[10] C. K. Chow. On Optimum Recognition Error and Reject Tradeoff. *IEEE Transactions on Information Theory*, 16(1), 1970.

[11] M. De Domenico, A. Lima, and M. Musolesi. Interdependence and Predictability of Human Mobility and Social Interactions. *Pervasive and Mobile Computing*, 9(6), 2013.

[12] T. M. T. Do and D. Gatica-Perez. Where and What: Using Smartphones to Predict Next Locations and Applications in Daily Life. *Pervasive and Mobile Computing*, 12, 2014.

[13] P. Domingo and M. Pazzani. Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classier. In *Proceedings of the 13th International Conference on Machine Learning (ICML)*, 1996.

[14] S. Džeroski and B. Ženko. Is Combining Classifiers with Stacking Better than Selecting the Best One? *Machine Learning*, 54(3), 2004.

[15] T. Economist. Move over, Siri. 2013.

[16] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera. An Overview of Ensemble Methods for Binary Classifiers in Multi-class Problems: Experimental Study on One-vs-One and One-vs-All Schemes. *Pattern Recognition*, 44(8), 2011.

[17] M. Galar, A. Fernandez, E. Barrenechea, H. Bustince, and F. Herrera. A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-based Approaches. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 42(4), 2012.

[18] S. Hettich and S. Bay. The UCI KDD Archive. `http://kdd.ics.uci.edu/`, 1999.

[19] E. Horvitz. Principles of Mixed-Imitative User Interfaces. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 1999.

[20] B. S. Jensen, J. E. Larsen, K. Jensen, J. Larsen, and L. K. Hansen. Estimating Human Predictability from Mobile Sensor Data. In *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2010.

[21] D. Kim, J. Hightower, R. Govindan, and D. Estrin. Discovering Semantically Meaningful Places from Pervasive RF-Beacons. In *Proceedings of the 11th International Conference on Ubiquitous Computing (UbiComp)*, 2009.

[22] N. Kiukkonen, J. Blom, O. Dousse, D. Gatica-Perez, and J. Laurila. Towards Rich Mobile Phone Datasets: Lausanne Data Collection Campaign. In *Proceedings of the 7th ACM International Conference on Pervasive Services (ICPS)*, 2010.

[23] W. Kleiminger, C. Beckel, T. Staake, and S. Santini. Occupancy Detection from Electricity Consumption Data. In *Proceedings of the 5th ACM Workshop on Embedded Systems For Energy-Efficient Buildings (BuildSys)*, 2013.

[24] C. Koehler, N. Banovic, I. Oakley, J. Mankoff, and A. K. Dey. Indoor – ALPS: An Adaptive Indoor Location Prediction System. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, 2014.

[25] J. Krumm and A. J. B. Brush. Learning Time-based Presence Probabilities. In *Proceedings of the 10th International Conference on Pervasive Computing (Pervasive)*, 2011.

[26] J. Krumm and E. Horvitz. Predestination: Inferring Destinations from Partial Trajectories. In *Proceedings of the 8th International Conference on Ubiquitous Computing (UbiComp)*, 2006.

[27] M. Kuhn and K. Johnson. *Applied Predictive Modeling*. New York: Springer, 2013.

[28] N. Lane, L. Pengyu, L. Zhou, and F. Zhao. Connecting Personal-scale Sensing and Networked Community Behavior to Infer Human Activities. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp)*, 2014.

[29] J. Laurila, D. Gatica-Perez, I. Aad, O. Bornet, T.-M.-T. Do, O. Dousse, J. Eberle, and M. Miettinen. The Mobile Data Challenge: Big Data for Mobile Computing Research. In *Proceedings of the Mobile Data Challenge Workshop, in conjunction with the 10th International Conference on Pervasive Computing (Pervasive)*, 2012.

[30] M. Lin, W.-J. Hsu, and Z. Q. Lee. Predictability of Individuals' Mobility with High-resolution Positioning Data. In *Proceedings of the 14th ACM International Conference on Ubiquitous Computing (UbiComp)*, 2012.

[31] O. Malik. The Four Megatrends Powering Predictive Computing.

[32] J. McInerney, S. Stein, A. Rogers, and N. Jennings. Exploring Periods of Low Predictability in Daily Life Mobility. In *Proceedings of the Mobile Data Challenge Workshop, in conjunction with the 10th International Conference on Pervasive Computing (Pervasive)*, 2012.

[33] A. Monreale and F. Pinelli. WhereNext: A Location Predictor on Trajectory Pattern Mining. In *Proceedings of the 15th ACM SIGKDD International Conference on*

Knowledge Discovery and Data Mining (KDD), 2009.

[34] R. Montoliu, J. Blom, and D. Gatica-Perez. Discovering Places of Interest in Everyday Life from Smartphone Data. *Multimedia Tools and Applications*, 62(1), 2012.

[35] A. Nicholson and B. Noble. BreadCrumbs: Forecasting Mobile Connectivity. In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2008.

[36] A. Niculescu-Mizil and R. Caruana. Predicting Good Probabilities with Supervised Learning. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, 2005.

[37] J. Platt. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. *Advances in Large Margin Classifiers*, 10(3), 1999.

[38] D. Salber, A. K. Dey, and G. D. Abowd. Ubiquitous Computing: Defining an HCI Research Agenda for an Emerging Interaction Paradigm. *Georgia Tech GVU Technical Report GIT-GVU-98-01*, 1998.

[39] S. Scellato, M. Musolesi, C. Mascolo, V. Latora, and A. Campbell. NextPlace: A Spatio-Temporal Prediction Framework for Pervasive Systems. In *Proceedings of the 9th International Conference on Pervasive Computing (Pervasive)*, 2011.

[40] J. Scott, A. J. B. Brush, and J. Krumm. PreHeat: Controlling Home Heating Using Occupancy Prediction. In *Proceedings of the 13th ACM International Conference on Ubiquitous Computing (UbiComp)*, 2011.

[41] C. Shin, J.-H. Hong, and A. K. Dey. Understanding and Prediction of Mobile Application Usage for Smart Phones. In *Proceedings of the 14th ACM International Conference on Ubiquitous Computing (UbiComp)*, 2012.

[42] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási. Limits of Predictability in Human Mobility. *Science*, 327(5968), 2010.

[43] L. Song and U. Deshpande. Predictability of WLAN Mobility and its Effects on Bandwidth Provisioning. In *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2006.

[44] L. Vu, P. Nguyen, K. Nahrstedt, and B. Richerzhagen. Characterizing and Modeling People Movement from Mobile Phone Sensing Traces. *Pervasive and Mobile Computing*, 17, 2014.

[45] D. Wagner, A. Rice, and A. Beresford. Device Analyzer: Understanding Smartphone Usage. In *Proceedings of the 10th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*, 2013.

[46] X. Wu, V. Kumar, R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. Yu, Z.-H. Zhou, M. Steinbach, D. Hand, and D. Steinberg. Top 10 Algorithms in Data Mining. *Knowledge and Information Systems*, 14(1), 2007.

[47] Y. Xu, M. Lin, H. Lu, G. Cardone, N. Lane, Z. Chen, A. Campbell, and T. Choudhury. Preference, Context and Communities: A Multi-faceted Approach to Predicting Smartphone App Usage Patterns. In *Proceedings of the 2013 International Symposium on Wearable Computers (ISWC)*, 2013.

[48] T. Yan, D. Chu, D. Ganesan, A. Kansal, and J. Liu. Fast App Launching for Mobile Devices Using Predictive User Context. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (Mobisys)*, 2012.

[49] B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *Proceedings of the 18th International Conference on Machine Learning (ICML)*, 2001.

[50] B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2002.