TouchML: A Machine Learning Toolkit for Modelling Spatial Touch Targeting Behaviour

Daniel Buschek, Florian Alt
Media Informatics Group, University of Munich (LMU)
{daniel.buschek, florian.alt}@ifi.lmu.de

ABSTRACT

Pointing tasks are commonly studied in HCI research, for example to evaluate and compare different interaction techniques or devices. A recent line of work has modelled user-specific touch behaviour with machine learning methods to reveal spatial targeting error patterns across the screen. These models can also be applied to improve accuracy of touchscreens and keyboards, and to recognise users and hand postures. However, no implementation of these techniques has been made publicly available yet, hindering broader use in research and practical deployments. Therefore, this paper presents a toolkit which implements such touch models for data analysis (Python), mobile applications (Java/Android), and the web (JavaScript). We demonstrate several applications, including hand posture recognition, on touch targeting data collected in a study with 24 participants. We consider different target types and hand postures, changing behaviour over time, and the influence of hand sizes.

Author Keywords

Touch; Toolkit; Machine Learning; Gaussian Process

ACM Classification Keywords

H.5.2. Information Interfaces and Presentation (e.g. HCI): Input devices and strategies (e.g. mouse, touchscreen)

INTRODUCTION AND RELATED WORK

Targeting experiments help HCI researchers to assess quantitative aspects of interactions, such as speed-accuracy tradeoffs, error rates, and throughput. Participants are asked to point at targets on the screen. The resulting data is commonly analysed with Fitts' Law [5]. Bi et al. [3] refined this approach for targeting with the finger in touch interaction. Other work modelled targeting while typing on touchscreens [1, 7, 14]. A different line of research compensated systematic patterns of touch-to-target errors ("touch offsets") with a polynomial function applied across many mobile device users [6]. More flexible offset models were trained on user-specific recorded touch targeting histories [13]. Touch offsets were also explored to recognise users and hand postures [4], to reveal characteristic screen regions for describing individual touch behaviour [11], and to reduce typing errors [12].

IUI'15, March 29 - April 01 2015, Atlanta, GA, USA

http://dx.doi.org/10.1145/2678025.2701381

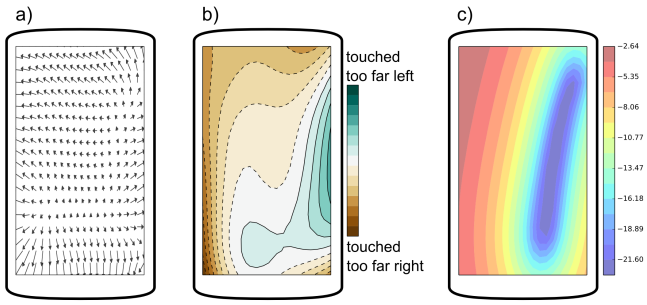

Figure 1. Three visualisations of Gaussian Process touch-to-target offset models: *a)* this arrow plot shows the model's 2D offset predictions across the screen, revealing the user's behavioural pattern for the right thumb when aiming at key-sized targets; *b)* the contour plot shows offset predictions for the same model in just one dimension (here: *x*) in colour; and *c)* shows the log-determinant of the predictive covariance of a model for full width targets, revealing the user's "arc of thumb reach".

These successful applications show the usefulness of modelling users' (individual) behaviour based on touch targeting data. In comparison to Fitts' Law, touch offset models ignore temporal information, but instead capture targeting error patterns across the screen. Hence, they provide an additional *spatial* perspective on user behaviour (Figure 1). However, this perspective is rarely explored, although the required touch and target locations are typically measured in most pointing experiments. In this paper, we present a toolkit which implements touch offset models to facilitate more widespread use in research and practical applications.

In particular, we contribute: 1) an introduction to touch offset models; 2) easy-to-use implementations and visualisations of these models for different use-cases (data analysis, practical deployment); and 3) example applications, including hand posture recognition, analysing more than 150.000 touches from a targeting experiment with 24 participants over two weeks. In contrast to previous studies addressing these models [4, 11, 13], we consider several target shapes and sizes, and also investigate the influence of different hand sizes.

TOUCH OFFSET MODELS

This section reviews touch offset models also employed in related work, introducing and describing linear [4] and non-linear [13] variants.

Touch Input

For the purpose of this paper and toolkit, we describe each touch $\mathbf{t}$ as a point on the plane, in other words as a two-dimensional vector, henceforth denoted as $\mathbf{t} = (x, y)^T \in \mathbb{R}^2$.

Linear Offset Models

A linear touch offset model with quadratic basis functions is described in related work [4]. Basis functions Φ transform the touch location $\mathbf{t}$ into a higher dimensional representation. With this trick, the linear model can account for non-linear relationships between touch and target locations across the screen, although it is still linear in its parameters. By default, the toolkit uses the same transformation Φ as the related work, namely $\Phi(\mathbf{t}) = (1, x, y, x^2, y^2)^T \in \mathbb{R}^5$.

Training

Training data consists of touches and corresponding target locations. Offsets are computed as the distances between touches and targets. The transformed touch locations $\Phi(\mathbf{t})$ of the N training examples are arranged as N rows in a design matrix $\mathbf{X} \in \mathbb{R}^{N \times 5}$. We then solve for the model's parameters $\mathbf{w}_x, \mathbf{w}_y$ with regularised linear regression:

$$\begin{aligned} \mathbf{w}_x &= (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{o}_x \\ \mathbf{w}_y &= (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{o}_y \end{aligned} \tag{1}$$

The vectors $\mathbf{o}_x$ and $\mathbf{o}_y$ contain the offsets along the x and y dimension of all training examples, respectively. $\lambda \in \mathbb{R}$ is the regularisation parameter. Regularisation helps to avoid overfitting (i.e. fitting noise in the training data).

Prediction

The weights $\mathbf{w}_x, \mathbf{w}_y$, together with the transformation Φ, define the linear touch offset model. For a new touch location $\mathbf{t}'$ the model predicts the offset μ:

$$\begin{aligned} \mu &= (\mu_x, \mu_y)^T \\ \mu_x &= \mathbf{w}_x^T\Phi(\mathbf{t}') \\ \mu_y &= \mathbf{w}_y^T\Phi(\mathbf{t}') \end{aligned} \tag{2}$$

Adding the predicted offset μ to the touch location $\mathbf{t}'$ yields a prediction for the true intended target location. Variances associated with these predictions are given by:

$$\begin{aligned} {\sigma'_x}^2 &= \hat{\sigma}_x^2\Phi(\mathbf{t}')^T(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\Phi(\mathbf{t}') \\ \hat{\sigma}_x^2 &= \frac{1}{N}\sum_{i=1}^{N}(o_x^{(i)} - \mathbf{w}^T\Phi(\mathbf{t}_i))^2 \end{aligned} \tag{3}$$

Analogously, ${\sigma'_y}^2$ and $\hat{\sigma}_y^2$ can be computed as well. In conclusion, the predicted information can define a bivariate Gaussian distribution $\mathcal{N}(\mathbf{t}' + \mu, \mathbf{\Sigma})$ with:

$$\mathbf{\Sigma} = \begin{bmatrix} {\sigma'_x}^2 & 0 \\ 0 & {\sigma'_y}^2 \end{bmatrix} \tag{4}$$

Non-Linear Offset Models

A non-linear modelling approach using Gaussian Process (GP) regression [9] was described by Weir et al. [13]. In contrast to the linear model, GPs offer a more flexible, non-parametric approach. In general, GPs are defined by a mean function μ and a covariance function C. We choose these parameters in line with the related work [13]: μ is set to constant zero. This means that the model predicts zero offsets without data. For C, we combine linear and squared exponential covariance functions, a flexible choice:

$$C(\mathbf{t}_m, \mathbf{t}_n) = a\mathbf{t}_m^T\mathbf{t}_n + (1-a)\exp(-\gamma\|\mathbf{t}_m - \mathbf{t}_n\|_2^2) \tag{5}$$

The mixture of linear and non-linear parts is controlled by a, and γ defines the length-scale of the Gaussian.

Training

We combine the training offsets from both dimensions into one long vector $\mathbf{z}$ and stack up the covariance matrix $\mathbf{C}_{mn} = C(\mathbf{t}_m, \mathbf{t}_n)$ of the training examples accordingly [13]:

$$\begin{aligned} \mathbf{z} &= [o_x^{(1)}, o_x^{(2)}, ..., o_x^{(N)}, o_y^{(1)}, o_y^{(2)}, ..., o_y^{(N)}]^T \\ \hat{\mathbf{C}} &= \begin{bmatrix} \mathbf{C} & \alpha\mathbf{C} \\ \alpha\mathbf{C} & \mathbf{C} \end{bmatrix} \end{aligned} \tag{6}$$

Here, α defines the dependence between x and y. We store $\mathbf{z}$ and the matrix $\mathbf{M} = (\hat{\mathbf{C}} + \sigma^2\mathbf{I})^{-1}$. Assuming noise variance σ^2 helps to map almost identical touches to different offsets.

Prediction

For a new touch location $\mathbf{t}'$, we create the vector $\mathbf{c}$, which contains all covariances of the new touch and the training touches. We then build its stacked up version $\hat{\mathbf{c}}$:

$$\begin{aligned} \mathbf{c} &= [C(\mathbf{t}', \mathbf{t_1}), ..., C(\mathbf{t}', \mathbf{t_N})] \\ \hat{\mathbf{c}} &= \begin{bmatrix} \mathbf{c} & \alpha\mathbf{c} \\ \alpha\mathbf{c} & \mathbf{c} \end{bmatrix} \end{aligned} \tag{7}$$

The model's prediction is a bivariate Gaussian $\mathcal{N}(\mu, \mathbf{\Sigma})$ with:

$$\begin{aligned} \mu &= \hat{\mathbf{c}}\mathbf{M}\mathbf{z} \\ \mathbf{\Sigma} &= \begin{bmatrix} C(\mathbf{t}', \mathbf{t}') & \alpha C(\mathbf{t}', \mathbf{t}') \\ \alpha C(\mathbf{t}', \mathbf{t}') & C(\mathbf{t}', \mathbf{t}') \end{bmatrix} - \hat{\mathbf{c}}\mathbf{M}\hat{\mathbf{c}}^T \end{aligned} \tag{8}$$

Adding the predicted mean offset μ to the sensed touch location $\mathbf{t}'$ can improve touch location. Moreover, $\mathcal{N}(\mathbf{t}' + \mu, \mathbf{\Sigma})$ gives the full predictive distribution.

THE TOOLKIT

A full documentation is provided with the toolkit. Here, we introduce its use in Python. The API style is inspired by the *scikit-learn* machine learning library [8], due to its ease-of-use. For efficiency, we use optimised *numpy* arrays [10].

Data Input

The toolkit uses a simple data format: N touches are given in a $N \times 4$ matrix, with columns *touch x, touch y, target x, target y*. This data could be loaded from a file or database, or collected and stored during runtime in an array. For example, we can load data from a csv-file with these four columns:

```
data = loadData("pointingStudy.csv")
```

As a result, *data* is a $N \times 4$ array containing the provided touch and target locations.

Creating and Applying Models

As a minimal example, creating and training offset models with default or custom hyperparameters could look like this:

```
model = LinearOffsetModel()
model.fit(data)

model2 = GPOffsetModel(gamma=2, noiseVar=0.001,
                       diag=0.9, kernelMix=0.1)
model2.fit(data)
```

After training, offset predictions for given touch locations can be computed as follows:

```
means, covs = model.predict(newTouches)
```

Here, *newTouches* is a $N \times 2$ array containing (new) touch locations. As a result, *means* is a $N \times 2$ array with the predicted offsets for these given touches (see μ in Equations 2 and 8). Complementary, *covs* is a list of N covariance matrices of shape 2×2 (see Σ in Equations 4 and 8).

These predictions can be used, for example, to correct the new touch locations and thus improve touch accuracy. As explained in the previous section, correction is performed by simply adding the predicted offsets to the touch locations:

```
correctedTouches = newTouches + means
```

Visualisation

The Python library also provides several visualisations to explore collected user data and plot the derived models. Figure 1 shows three examples for touch behaviour on a smartphone, visualising offset predictions and variances with arrows and contours. Plots can be created as follows:

```
arrowPlot(model)
contourPlot(model, dim="x")
variancePlot(model)
```

Implementation Aspects

We describe some aspects of our implementation to highlight the benefits of using the toolkit compared to other options:

Optimisation: Computing many predictions is important for data analysis tasks. For instance, consider ten-fold cross-validation to evaluate improving touch accuracy (e.g. Figure 2). On our dataset, such an analysis takes 109s with an implementation of the GP "textbook formulas" - but 18s with the toolkit (measured on i7/2.8Ghz PC, 8GB RAM), achieved by rearranging computations to better utilise matrix routines [10].

Specialisation: Our GP implementation also respects possible dependencies between x and y locations, following related work [13]. Less specialised libraries may not provide these models at all, or they do not implement this special GP variant (e.g. [8]) employed for touch modelling [13]. Moreover, our toolkit's visualisations allow for fast exploration and assessment of the models and the captured behavioural patterns.

Mobile applications: Besides Python, the toolkit provides models in Java/Android and JavaScript, respecting some platform limitations. For example, the models can be used to improve touch accuracy in apps and on websites. Detailed descriptions and examples are provided with the toolkit.

EXPERIMENT AND EXAMPLE APPLICATIONS

As examples for using the toolkit, this section presents three applications to data collected in a targeting experiment.

Dataset

We collected a dataset with over 150.000 touches from 24 participants in a targeting experiment on a Nexus 5 phone. Each participant completed 8 tasks (2 hand postures $\times$ 4 target types). Each task showed 400 targets distributed equally across the screen, displayed one at a time in random order.

The two hand postures were: 1) *thumb*, holding the phone in the right hand, touching with the right thumb; and 2) *index finger*, holding it in the left hand, touching with the right index finger. The phone was always held in portrait orientation. The four target types were (see Figure 2): *crosshair*, *key-sized* (4×7mm), *app-sized* (9×9mm), and *full width* (height 9mm). Task order was varied between participants with a 8×8 latin-square design. Each participant repeated all tasks in a second session one week later.

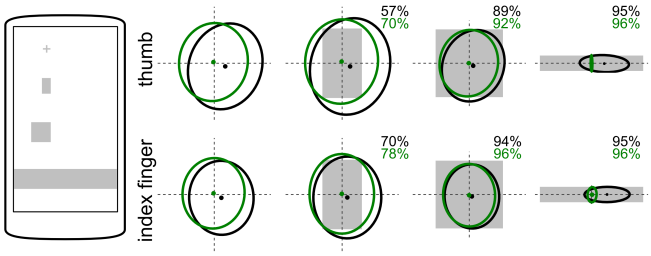

Figure 2. Targets used in the study, shown to scale (left). The plots to the right summarise the distributions of touches from our dataset around these targets for raw touches (black) and touches corrected with offset models (green). The models shifted touches towards the target centres, resulting in a higher overlap of touch distributions and target bounding boxes. Hence, users can hit their intended targets more reliably with offset model corrections (hit rates annotated for targets with an area).

In this note, we focus on three analyses of this dataset to show different uses of offset modelling: 1) improving accuracy, 2) analysing the influence of hand size on targeting behaviour, and 3) predicting hand postures.

Improving Touch Accuracy

In this first application, we address improving touch accuracy by correcting touch locations with the models' predictions. We trained linear offset models on the data from the first session of each user, and then applied these models to correct the touches of each user collected in the second session.

Figure 2 summarises the results, showing distributions of raw and corrected touches around the different targets for both hand postures. Offset models correct touches towards the target centres, improving hit rates.

Analysing the Influence of Hand Size

In our experiment, we also measured participants' hand sizes, using images captured by placing their hands on a scanner. In particular, we measured the distance from the tip of the index finger to the bottom of the palm (Lunate bone). In this way, we obtained hand sizes between 159mm and 194mm.

In this analysis, we are interested in the influence of hand sizes on targeting patterns. In particular, we expect larger y-offsets near the top (left) of the screen with thumb input, since this region is harder to reach on a larger device such as ours (137.8mm $\times$ 69.1mm). Sorting participants by hand size revealed three groups. We focus on the two extremes: small-handed users (N=4, range 159mm to 171mm) and large-handed ones (N=7, range 186mm to 194mm). Using our toolkit, we trained GP models on each groups' thumb touches aiming at *crosshair*, *key-sized*, and *app-sized* targets.

Figure 3 visualises the differences in the lengths of the models' predicted y-offsets: Users with small hands indeed showed longer y-offsets in the upper left corner, but are more

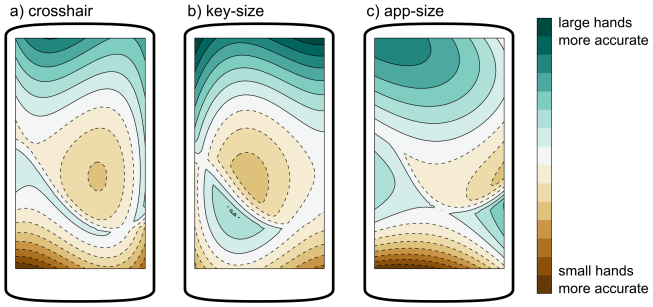

Figure 3. Differences in lengths of the predicted y-offsets of two GP models (for thumb targeting), one representing small-handed users, the other large-handed ones. These plots show that larger hands result in targeting patterns with shorter y-offsets near the top of the screen. In contrast, smaller hands are more accurate near the bottom and centre.

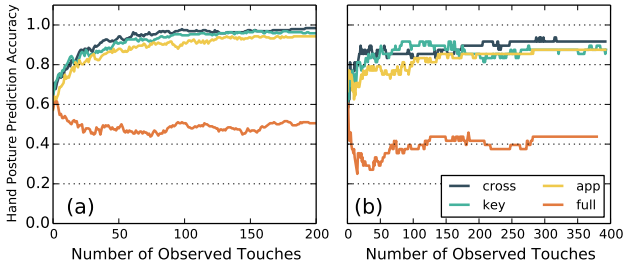

Figure 4. Hand posture prediction accuracy. We deployed offset models to predict hand postures (thumb vs index finger). Our approach achieved 94-98% accuracy within one session *(a)* and 87-92% across sessions *(b)*, for three of four target types. Prediction was not feasible for *full width* targets, indicating little posture-specific information in offset patterns for this target type.

accurate near the bottom of the screen. We also found interesting regions to the bottom left of the centre, where users with large hands were more accurate along the y-dimension. This may indicate an area of comfortable bend for larger thumbs. A future study could investigate this further, for example by filming participants' hands during interaction.

Predicting Hand Postures

We implemented a hand posture recogniser to predict the currently used hand posture after each touch. The general approach follows related work [4], but uses non-linear GP models from our toolkit, joint distributions to accumulate evidence over time, and different target types. Intuitively, we match observed offsets with predictions from one model per posture to decide for the posture whose model provides the best explanation for the observed targeting behaviour.

We evaluated this approach for two cases: *within sessions* - training (testing) models with the first (second) half of the data from one study session; and *across sessions* - training models on data from one session and testing them with touches from the second session collected one week later.

Figure 4 shows that our system predicts the correct hand posture with high accuracy (within: 94-98%, across: 87-92%). It reached >80% after 25-100 touches. These accuracies are comparable to the results from related work on typing posture detection [14]. Predictions were not successful for *full width* targets, which indicates that the corresponding offset patterns were less posture-specific than for the other targets.

DISCUSSION

We discuss the models regarding generalisation and usage scenarios, along with consequences of incorrect predictions:

Generalisation: "In the wild", behaviour will vary more (e.g. different grip, devices, walking), leading to less precise predictions, if the model is not trained on such data. However, related work has shown that offset patterns to some extent transfer across users and devices [4], and that models can improve accuracy even when trained across many users/devices with data collected in a large "in the wild" study [6].

Model robustness: From related work [4, 6], and the investigations of this paper, we find that simpler models seem practically more robust. On the other hand, complex non-linear models can more accurately capture details of behaviour in stable conditions. This can be preferable, for example, when studying behaviour observed in a controlled lab experiment.

Prediction errors: If predictions are not accurate, the touch location might not be shifted directly towards the target. How serious this case is depends on the application. While improving touch accuracy obviously requires accurate predictions, a few imprecise predictions may be acceptable if we can integrate over many touches (e.g. for hand posture detection).

Usage scenarios: Besides the applications shown here, models could help to distinguish users based on touch behaviour for continuous implicit authentication. They could also inform GUI layouts - place important target elements in the user's "comfort zone" of highest touch accuracy. The layout could be dynamically tailored to the individual user and context, as captured by a continuously updated offset model.

CONCLUSIONS AND FUTURE WORK

Targeting experiments help researchers to evaluate and compare different interaction techniques or devices. We have presented a toolkit and example applications for touch offset modelling. These models capture targeting error patterns across the screen, providing a different spatial perspective on user behaviour, which cannot be assessed with more common evaluations, such as Fitts' Law. We have applied the toolkit on a large targeting dataset to improve touch accuracy, to analyse the influence of hand size on targeting patterns, and to predict hand postures. We hope that the presented toolkit facilitates more widespread use of spatial touch modelling.

To the best of our knowledge, we are the first to relate touch offset models to measured hand sizes. Future work could investigate this further, for example to predict the user's hand size from touch offsets, possibly combining it with models for thumb reach from related work [2]. Predicting postures is feasible, as shown in this paper for different target types. We aim to deploy and evaluate the hand posture recogniser in a typical mobile interface, such as a homescreen or keyboard.

Although we have presented the toolkit with touch interactions in mind, it may also be applied to other input methods in which users aim at targets on a two-dimensional surface (e.g. mouse cursor, stylus interaction).

PROJECT RESOURCES

The toolkit, documentation, and study data are available at:
`http://www.medien.ifi.lmu.de/touchml/`

REFERENCES

1. Baldwin, T., and Chai, J. Towards Online Adaptation and Personalization of Key-Target Resizing for Mobile Devices. In *Proceedings of the 17th International Conference on Intelligent User Interfaces* (2012), 11–20.
2. Bergstrom-Lehtovirta, J., and Oulasvirta, A. Modeling the Functional Area of the Thumb on Mobile Touchscreen Surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2014), 1991–2000.
3. Bi, X., Li, Y., and Zhai, S. FFitts Law: Modeling Finger Touch with Fitts' Law. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2013), 1363–1372.
4. Buschek, D., Rogers, S., and Murray-Smith, R. User-Specific Touch Models in a Cross-Device Context. In *Proceedings of the 15th International Conference on Human-Computer Interaction with Mobile Devices and Services* (2013), 382–391.
5. Fitts, P. M. The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. *Journal of Experimental Psychology 47*, 6 (1954), 381–391.
6. Henze, N., Rukzio, E., and Boll, S. 100,000,000 Taps: Analysis and Improvement of Touch Performance in the Large. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services* (2011), 133–142.
7. Mock, P., Edelmann, J., Schilling, A., and Rosenstiel, W. User Identification Using Raw Sensor Data From Typing on Interactive Displays. In *Proceedings of the 19th International Conference on Intelligent User Interfaces* (2014), 67–72.
8. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res. 12* (Nov. 2011), 2825–2830.
9. Rasmussen, C. E., and Williams, C. K. I. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.
10. van der Walt, S., Colbert, S., and Varoquaux, G. The NumPy Array: A Structure for Efficient Numerical Computation. *Computing in Science Engineering 13*, 2 (March 2011), 22–30.
11. Weir, D., Buschek, D., and Rogers, S. Sparse Selection of Training Data for Touch Correction Systems. In *Proceedings of the 15th International Conference on Human-Computer Interaction with Mobile Devices and Services* (2013), 404–407.
12. Weir, D., Pohl, H., Rogers, S., Vertanen, K., and Kristensson, P. Uncertain Text Entry on Mobile Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2014), 2307–2316.
13. Weir, D., Rogers, S., Murray-Smith, R., and Löchtefeld, M. A User-Specific Machine Learning Approach for Improving Touch Accuracy on Mobile Devices. In *Proceedings of the 25th annual Symposium on User Interface Software and Technology* (2012), 465–476.
14. Yin, Y., Ouyang, T. Y., Partridge, K., and Zhai, S. Making Touchscreen Keyboards Adaptive to Keys, Hand Postures, and Individuals - A Hierarchical Spatial Backoff Model Approach. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2013), 2775–2784.