Making Touchscreen Keyboards Adaptive to Keys, Hand Postures, and Individuals - A Hierarchical Spatial Backoff Model Approach

Ying Yin[*]
yingyin@mit.edu

Tom Y. Ouyang[†]
ouyang@google.com

Kurt Partridge[†]
kep@google.com

Shumin Zhai[†]
zhai@google.com

[*]Massachusetts Institute of Technology
32 Vassar St.
Cambridge, MA U.S.

[†]Google Inc.
1600 Amphitheatre Parkway
Mountain View, CA U.S.

ABSTRACT

We propose a new approach for improving text entry accuracy on touchscreen keyboards by adapting the underlying spatial model to factors such as input hand postures, individuals, and target key positions. To combine these factors together, we introduce a hierarchical spatial backoff model (SBM) that consists of submodels with different levels of complexity. The most general model includes no adaptive factors, whereas the most specific model includes all three. Considering that in practice people may switch hand postures (e.g., from two-thumb to one-finger) to better suit a situation, and that the specific submodels may take time to train for each user, a specific submodel should be applied only if its corresponding input posture can be identified with confidence, and if the submodel has enough training data from the user. We introduce the *backoff* mechanism to fall back to a simpler model if either of these conditions are not met. We implemented a prototype system capable of reducing the language-model-independent error rate by 13.2% using an online posture classifier with 86.4% accuracy. Further improvements in error rate may be possible with even better posture classification.

Author Keywords

Touchscreen text input; posture adaptation; personalization; adaptive model.

ACM Classification Keywords

H.5.2. Information interfaces and presentation: User interfaces.

INTRODUCTION

The rapid growth of touchscreen based smartphones and tablets have made finger typing on touchscreens an everyday information input activity. Touchscreen keyboards, which can also be called Smart Touch Keyboards (STKs), have advanced significantly in the past few years. Taking the open source Android keyboard as an example, a modern touch screen keyboard uses language modelling, spatial and edit distance based corrections, and other sophisticated techniques to predict, correct, and complete the users imprecise typing. Despite these engineering achievements, text input continues to be a mobile user experience bottleneck, particularly for business and productive use [3]. Any further improvement to the mobile typing experience, even by a small amount, is desired and important since hundreds of millions of people use a smartphone or tablet everyday.

One compelling direction of research is better adaptation and personalization of keyboard spatial models. A spatial model converts each touch point into a probability distribution over the different letters on the keyboard. With soft keyboards, the underlying "keys" can shift and adapt to the user. Azenkot and Zhai [2] did a systematic study of smartphone keyboard touch patterns under various typing conditions. We note the following observations from their study:

1. People use different "hand postures" - one index finger, one thumb, and two thumbs - to type on smartphones. Depending on the situation (e.g. sitting, standing or walking), the same individual may also change from one hand posture to another. For example, the same individual who normally types with two thumbs on a phone while sitting down may switch to one index finger or one thumb typing while standing or walking. We cannot assume that the same individual will use the same hand posture all the time. Adaptation methods based on lab experiments with one consistent hand posture, such as those in [6], give important insights and guidance for designing practical systems. But they may also face challenges as practical solutions because user may change their hand posture in real world settings from their usual preference.

2. These hand postures change the touch typing patterns. For example, for right handed users adopting one index finger and one thumb typing postures, touches on the left side of a smartphone touchscreen keyboard tend to be biased rightward. However these touches tend to be biased leftward among users adopting two thumb postures.

CHI 2013, In proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pages 2775 - 2784, 2013.

3. Touch patterns can also depend on letter keys. For example, for one-finger typing, touch points tend to shift downward on the bottom letter row of a touchscreen keyboard, but not the top row (Q - P).

4. Users touch points tend to spread wider on a collective basis (polling all users data together) than on an individual basis for the same posture. This shows that different users tend to have different touch point distributions, but within the same user, the touch point pattern using the same posture may be more consistent.

Together these findings make a strong case for personalization of smartphone touch keyboard algorithms. However they also illustrate the challenges and complexities of personalization. An advanced keyboard that adapts to user differences needs to account for multiple *adaptive factors* in order to work effectively. For example, a personalized typing model would have limited effectiveness if it could not also adapt to hand posture. This is because the same user can have a very different typing pattern depending on whether she is typing with one finger or two. Effective adaptation may require a combinatorial approach that includes key location, hand posture, and user personalization.

A combinatorial approach raises challenging implementation issues. First, there need to be a large number (e.g., 26 keys $\times$ 3 postures = 78) of submodels for each user. Collecting a sufficient amount of data to build each submodel may take too long to be practically useful, especially for infrequent letters such as, in English, "Z" and "X." Second, if an STK does have a large number of (sub)models, model selection can be a significant challenge. Since each submodel is specific to a combination of factors, a wrong selection may actually hurt the keyboards quality. Correct model selection requires an accurate identification of the current *mode*, i.e. a combination of adaptive factors. While it is relatively easy to identify the individual (for example, by device login), it is not so easy to identify what hand posture the user is applying to each tap.

To address these challenges, we propose and explore a hierarchical backoff approach to building adaptive STK spatial models. The models in the hierarchy range from generic (e.g. a base model that is user, key, and posture invariant) to specific (e.g., a model that has specialized parameters for each user, key, and posture combination). When a user first starts typing on a hierarchical adaptive keyboard, it initially defaults to the base model, and the other more specific models are dormant. The system continues to collect touch points data from the user to train submodels. A submodel becomes *mature* when a sufficient amount of data is available to train it. Determining the threshold for the minimum amount of data required is an essential part of the backoff model, which we discuss later in the paper. A *mature* model will continue to renew itself with new user data to accommodate both short and long-term adaptation to changes in user behavior. If there is not enough data to train the model or the system is not confident enough that the model is appropriate, the system *backs-off* to a more general model or the base model. This hierarchical backoff approach is a major contribution of this paper, and offers the following advantages for fast and robust adaptation in an STK:

1. It does combinatorial and fine grained adaptation, not only to the user as a single lumped entity, but also in combination with other factors such as hand posture and key location. It is therefore more practical and less brittle since it does not assume one user will always use the same hand posture in all situations.

2. It is conservative. Posture adaptation is applied only if the posture detection is confident enough, and when its corresponding submodel is *mature*. It is also designed to be conservative and biased towards the standard base model if either of these condition is not clearly satisfied. This minimizes the risk of over-adaptation and transitional instability when the user changes hand posture.

3. The system does not require a separate training (data collection) phase for each individual. Instead, it adapts to a user gradually as she is typing. This makes it more practical to deploy and eliminates any additional interruption or burden on the user.

4. The system continually updates and renews itself, so it can accommodate long-term changes in user behavior.

We call keyboards built in such an approach SBM (spatial backoff model) keyboards. The SBM approach also raises many questions and challenges that will be addressed in the rest of the paper. Here we give a brief outline to these problems and their solutions.

We first report one analysis on the relative key detection power of various spatial models specific to the combinatorial factors of keys, hand postures, and users in the "Comparison of Spatial Model" section. The analysis is based on *observed* posture and user-intended key. This means the true identity of the posture and the keys users intend to type are known. With our dataset it was found that, compared to the base model, posture- and key-specific spatial models could reduce character error rate by about 11.5%, and that user- and key-specific spatial models could reduce character error rate by about 14.2%. These reductions in error rates represent theoretical upper bounds based on our dataset.

Second, in the "Input Hand Posture Classification" section, we present a detailed SVM (support vector machine) classifier of hand postures that distinguishes between two-thumb and one-finger (including one-thumb and one-index-finger) postures from the user's touch points on the fly while she is entering text. This approach is able to achieve an accuracy rate of 86.4%.

Third, in the "Evaluation of Standalone SBM" section, we show that posture classification combined with adaptive models improves spatial model performance, reducing the character error rate by 13.2% over a non-adaptive baseline model.

RELATED WORK

There is a body of active research on using spatial models, language models, and a combination of the two to improve text entry accuracy on an STK. For example, Al Faraj et al. [1] and Magnien et al. [16] both use visual highlight of the next possible keys to aid typing. Their predictions of the next keys are based on a language model which is independent of the improvements we propose through spatial model adaptation.

Kristensson et al. [13] propose a geometric pattern matching technique to improve stylus input accuracy. They match the geometric pattern of the touch points on a stylus keyboard against patterns formed by the letter key center positions of legitimate words in a lexicon. Their approach uses a combination of spatial and language models. However, their spatial model is not adaptive. While our focus is on tapping input, the same adaptive spatial model approach can be potentially applied to the gesture-based input.

Goodman et al. [7] explore key adaptive spatial models for stylus input by using separate bivariate Gaussian distributions with means and covariance matrices per key. Zhai et al. also show that the hit points for each key on a custom keyboard [20] are normally distributed, but furthermore that the centers of the distributions shift in different directions depending on the positions of the keys [21]. In their relative keyboard input system, Rashid et al. [17] use a different bivariate Gaussian for each key. However, the error rate of their system is high because of the keyboard position is not fixed.

Building on [7], Gunawardana et al. [8] use restricted bivariate Gaussian models in their anchored key-target resizing method. Key-target resizing means dynamically adjusting the underlying target areas of the keys based on their probabilities. The probabilities can be a combination of spatial model and language model probabilities. They argue that overly aggressive key-target resizing can sometimes prevent a user from entering their desired text, and hence violate his or her expectation about keyboard functionality. They ensure that a touch point within the anchor area of a key is always detected as that key irrespective of the language model.

In personal adaptation for STK, Findlater and Wobbrock [6] explore spatial model adaptation on large touch surfaces, in which one can use ten fingers to do traditional desktop style touch typing. Based on a user study of 12 participants, they find measurable performance improvements when the keyboard touch typing model is able to adapt to a particular user.

In comparison to ten-finger typing on a large touch surface, the individual differences in one-finger or two-thumb typing on smartphones are more subtle but still compelling. Rudchenko et al. [18] developed a text entry game for smartphones that provides targeting words for users to type to improve their typing experience. As a side effect, the game generates labeled touch point data that can be used as training data to build spatial models. Their results, based on a user study of 6 participants, show that key-target resizing based on a spatial model only, without personalization, gives an error reduction of 18.9% over no key-target resizing. When adding personalization, there is a further 2.84% error reduction. The results show the benefit of user adaptation in an ideal condition in which the intended key is known. In real use, however, the intended key is unknown and can only be inferred from the current spatial and language models. In our prototype implementation and evaluation we impose this same real-world limitation. We build the user-adaptive spatial model by probabilistically assigning a key to each of the user's touch points, without relying on the hidden identity of the true intended key.

The user- and key-adaptive methods mentioned above all assume that a user's input posture remains the same. As far as we are aware, our work is the first to investigate adaptation of dynamically changing postures.

There is also a body of applicable work on user modeling and adaptation not directly related to touch keyboard typing. A system can be considered a user-adaptive system if it makes nontrivial inferences about properties of the user, and adapts its actions to these inferred properties [10]. Some systems use rule-based adaptation that may lack empirical justification, while others use decision-theoretic methods (e.g., [11]). Jameson et al. [10] argue that it is important to develop decision procedures for adaptation in a principled and empirically justified way, and they present a method to do so based on a Bayesian network model. We also use a data-driven approach for adaptation and build a relative simple user model compared to theirs. More work could be done in the user-modeling direction to incorporate even more properties for STK adaptation, e.g., whether the user is walking or sitting, the user's body posture, etc.

The idea of backing off to a lower level model when a more specific model is not available is commonly used in language modeling for speech recognition [12, 22]. For example, when unseen n-gram events are encountered, the backoff class-based n-gram language model is used. In language modeling, all variables are observed including backoffs. However in our model some of the variables are observed, e.g. user, whereas some are hidden and need to be inferred, e.g. posture. As a result, the backoff conditions are more complicated in our model.

Finally, researchers have explored other ways to improve the text input experience on touchscreen keyboards. One method is to vibrotactile feedback [5, 9]. Another is to use alternative keyboard layouts optimized for typing speed (word per minute) based on Fitts' law and character level digraph frequencies [20, 15]. These dimensions are orthogonal to the language and the spatial models, and can be potentially combined together to further increase the input accuracy and speed.

RESEARCH METHODS

We use a combination of HCI and ML (machine learning) methods to develop and evaluate the SBM.

We use a previously published smartphone typing dataset as the empirical basis for our training and cross-validation [2]. We will call it the Pepper dataset in the rest of the paper for easy reference. Briefly, the experiment involved 32 participants who were given random phrases to type on a "data

collector" keyboard on an Android touchscreen phone. The "data collector" keyboard was designed to collect data that reflected users' natural instincts, uninfluenced by the keyboard's actual performance. The keyboard therefore displayed only an asterisk as a placeholder for each tap, and did not provide a backspace key. The experiment was between-subjects, with each subject adopting a different hand posture. For consistency, we removed the data from the two left-handed users in our analysis. This leaves 9 users using one-index-finger input, 11 users using one-thumb input, and 10 users using two-thumb input. We also filtered out touch points that are 1.5 times the height of the key away from the center of the target key. After this, there are 84,292 total touch points.

A basic dependent variable of the dataset and any models built from it is "character error rate." This is measured by the difference between the character in the target phrase and character determined from the touch point via a spatial model. Note that although this rate can be viewed as a percentage of the total number of characters in the target phrase, it is to be interpreted cautiously. First, a 0% error rate may never be achievable in the Pepper dataset because the participants were asked to type naturally and fast. Second, the absolute error rate maybe dataset-dependent and is not necessarily what a user would experience in practice. Nonetheless, the comparative error rates across conditions can be informative of the qualities of different models.

To get a sense of the error variation among different input postures, we calculate the mean error rates by checking whether each touch point lies within the bounding box of the target key. The mean error rates per person using this method are 7.98% (SD = 5.7) overall, 7.92% (SD = 5.8) for one-index-finger, 6.59% (SD = 3.7) for one-thumb, and 10.69% (SD = 7.4) for two-thumb input respectively. Since there is much less difference between the index finger posture and the single thumb posture, and because of posture classification reliability concerns, we combine the index-finger posture and the one-thumb posture into a single one-finger posture condition. Hence the posture class random variable y can take values in the set $\mathcal{Y} = \{\text{one-finger}, \text{two-thumb}\}$. For all evaluations below, the ratio of the number of users using one-finger input to that using two-thumb input is kept the same for the training and testing datasets.

HIERARCHICAL SPATIAL BACKOFF MODELS (SBM)

The hierarchical adaptive SBM consists of a number of submodels in different "levels" (Figure 1). Each submodel is represented by a bivariate Gaussian distribution [2, 7, 17]. The lowest level is the base model which is key-, user-, and posture- independent, i.e. all the keys have the same Gaussian distribution $N(\underline{\mu}, \Lambda)$ where $\underline{\mu} \in \mathbb{R}^2$ is the mean (x, y) offsets from the center of each key's bounding box, and Λ is the 2×2 covariance matrix. This is the most general model combining all data together. The higher level submodels adapt to a combination of key positions, hand postures, and users. For example, the "posture-adaptive" model adapts to input posture only, which means there are separate Gaussian models $N(\underline{\mu}_y, \Lambda_y)$ for each posture y for *all* keys. The "posture- & key-adaptive" model adapts to both posture and key which means there are separate Gaussian models $N(\underline{\mu}_{y,k}, \Lambda_{y,k})$ for each posture y and key k combination. At the highest level there are separate Gaussian models $N(\underline{\mu}_{y,u,k}, \Lambda_{y,u,k})$ for each posture y, user u, and key k combination.

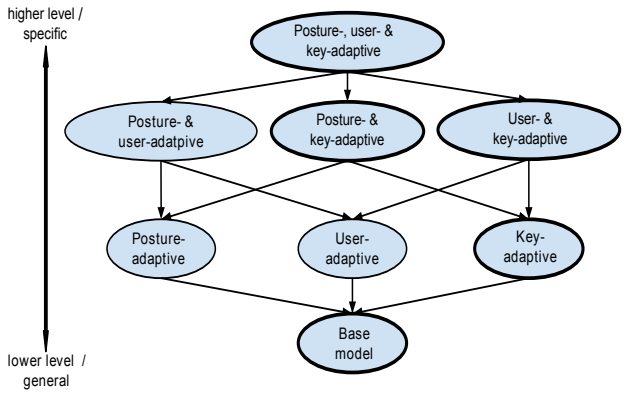


Figure 1. A complete hierarchical spatial backoff model. For practical purposes, not all the submodels need to be included. The models with thicker lines are the ones we included in the prototype implementation.

Each combination of the factors needs a sufficient number of samples to build a reliable model. Hence, each submodel would only become *mature* when its reliability passes a set threshold. Otherwise a lower order model (backoff) will be used instead.

Figure 1 shows a *complete* hierarchy of the submodels with all possible combinations of the three adaptive factors. However, depending on the relative effectiveness of the submodels, it may not be necessary to include all of them in an implementation. In this paper, we focus on analyzing the key-adaptive model, posture- and key-adaptive model and user- and key-adaptive model. The order of the backoff process, and the priority of the submodels at the same level can also be design choices, but the analysis in the next section gives guidance and suggestions on how to determine an order.

KEY ESTIMATION FORMULATION

As mentioned in the Research Methods section, our basic measure for analysis and comparison of the spatial models is character error rate. This is computed as:

$$\text{character error rate} = \frac{\text{\# wrongly estimated characters}}{\text{\# all target characters}} \quad (1)$$

Character estimation is made based on an underlying model, which can consist of both a language model and a spatial model. We use $\underline{\theta}$ to denote the parameter vector of the overall model. Given i^{th} touch point coordinates $\underline{c}_i \in \mathbb{R}^2$, we estimate the input character as the most likely intended key, $\hat{k}_i$,

given by the following formulation:

$$\hat{k}_i = \arg\max_k p(k|\underline{c}_i;\underline{\theta}) \quad (2)$$

$$= \arg\max_k \frac{p(\underline{c}_i,k;\underline{\theta})}{\sum_k p(\underline{c}_i,k;\underline{\theta})} \quad (3)$$

$$= \arg\max_k p(\underline{c}_i,k;\underline{\theta}) \quad (4)$$

$$= \arg\max_k p(k;\underline{\theta}_l)p(\underline{c}_i|k;\underline{\theta}_s) \quad (5)$$

where $\underline{\theta}_l$ represents the parameter vector of the language model and $\underline{\theta}_s$ represents the parameter vector of the spatial model. The $p(k;\underline{\theta}_l)$ term is related to the frequencies of letters and is part of the language model. To investigate whether spatial model adaptation can be beneficial, we analyze key estimation using the spatial model alone as a first step. Hence we assume $p(k;\underline{\theta}_l)$ is the same for all k (i.e. a uniform language model). As a result

$$\hat{k}_i = \arg\max_k p(\underline{c}_i|k;\underline{\theta}_s) \quad (6)$$

The spatial model parameter vector $\underline{\theta}_s$ depends on the sub-model selected from the hierarchical SBM. For example, when the posture-, user- and key-adaptive model is selected, then

$$\hat{k}_i = \arg\max_k \sum_{y,u} N(\underline{c}_i - \underline{o}_k; \underline{\mu}_{y,u,k}, \Lambda_{y,u,k})[[y = \mathsf{y} \wedge u = \mathsf{u}]] \quad (7)$$

where $\underline{o}_k$ is the center of the visual bounding box of key k, and $\underline{c}_i - \underline{o}_k$ represents the offset. y and u represent a particular value for the posture and the user respectively. $[[y = \mathsf{y} \wedge u = \mathsf{u}]] = 1$ if the statement is true, and 0 otherwise. Note that we use serif fonts, e.g., y, to denote random variables, and use sans-serif fonts, e.g., y, to denote known quantities.

The above formulation assumes the values of the posture and user variables are known. For the user variable, this is reasonable because smartphones are personal devices and we can assume that the same user types on the same device. For the posture variable, potentially we could use a soft decision and combine probability distributions together based on the posture probability, $p(y)$. For the analysis in the "Comparison of Spatial Models" section below, the values for y and u are *observed* based on the dataset. This establishes the upper bound for the potential benefits from the adaptations. In the "Implementation and Evaluation" section, only the user is *observed*; the value for the posture variable y is *inferred* through probabilistic classification.

COMPARISON OF SPATIAL MODELS

We compare character error rate with different adaptive spatial models to analyze their relative effectiveness (Table 1 and Figure 2)). This can suggest the order of the backoff models to use when there is not enough data for higher level models. 10-fold cross-validation is used, and the training and testing data sets have different users.

The simplest model is a Gaussian distribution with zero mean and the same spherical covariance matrix for all the keys.

Spatial model	**Character error rate (SD)**
Distance from the center of the keys	7.98% (5.7)
Base model (same Gaussian model $N(\underline{0},\Lambda)$ for all the keys with a full covariance matrix)	7.85% (5.4)
Key-adaptive model, $N(\underline{\mu}_k,\Lambda_k)$	8.02% (5.6)
Posture & key-adaptive model, $N(\underline{\mu}_{y,k},\Lambda_{y,k})$	7.06% (4.4)
Individual & key-adaptive model, $N(\underline{\mu}_{u,k},\Lambda_{u,k})$	6.85% (4.4)

Table 1. Comparison of character error rate (no language model) with different spatial models using 10-fold cross-validation. These results represent the theoretical upper bounds for the different adaptive spatial models based on the Pepper dataset since they assume *observed* postures and intended keys when building the models.

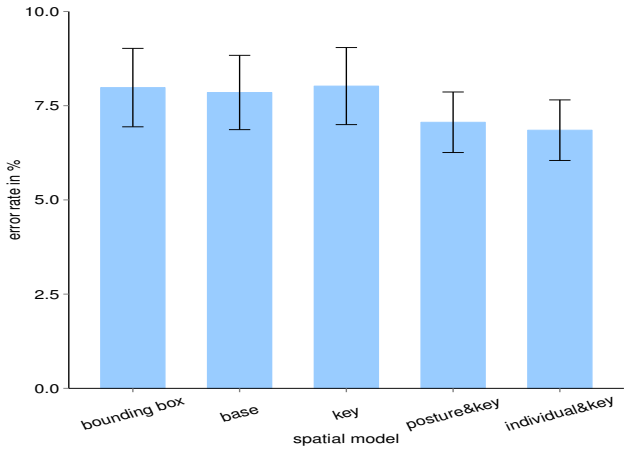


Figure 2. Bar plot of error rates in Table 1 with error bars indicating the standard error of the mean (SEM).

This model detects keys by choosing the key with the shortest Euclidean distance from the tapping coordinates. Our base model has a full covariance matrix Λ learned from the training data, but the same base model $N(\underline{0},\Lambda)$ is used for all keys. Hence, when using the base model for key estimation, the most likely intended key is

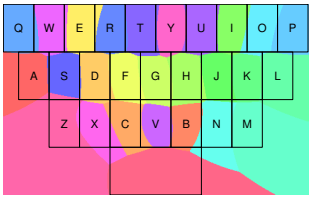
$$\hat{k} = \arg\max_k N(\underline{c}_i - \underline{o}_k; \underline{0}, \Lambda) \quad (8)$$

Key Adaptation

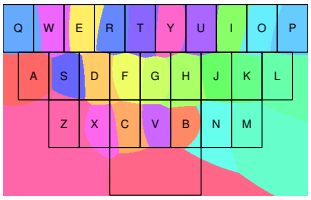
A basic key-adaptive model has one bivariate Gaussian model $N(\underline{\mu}_k,\Lambda_k)$ for each key k built using data from all users in the training dataset. The most likely intended key based on key-adaptive model is

$$\hat{k}_i = \arg\max_k N(\underline{c}_i - \underline{o}_k; \underline{\mu}_k, \Lambda_k) \quad (9)$$

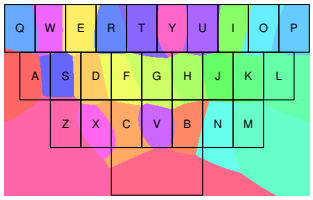
The result in Table 1 shows that the key-adaptive model has a higher error rate than the base model. A little investigation shows that different hand postures tend to shift key specific spatial model in different ways, sometimes even in opposite directions. For example, in Figure 3(a), we observe that the



(a) Key-adaptive model



(b) Posture- and key-adaptive model for one-finger input



(c) Posture- and key-adaptive model for two-thumb input

Figure 3. Comparison of effective key areas with different spatial models. Each colored area presents the region such that if the user taps in that region, the spatial model will classify that key with the corresponding key label.

effective area of the "W" key goes slightly beyond the visual key boundary between "E" and "W." For one-finger input, there is a horizontal offset biased to the right for the "W" key, whereas for two-thumb input, there is a horizontal offset biased to the left for the "W" key. When mixing these opposite effects together in key adaptation, no meaningful results could be expected. The base model suffers the same setback, but even though neither of the two models closely models the underlying distributions of the touch points, the base model, with fewer parameters than the key-adaptive model, may be more generalizable when there are limited data.

Posture Adaptation

Key adaptation becomes more effective when combined with posture adaptation as shown in Table 1. There is 12.0% reduction in error rate compared to key adaptation only. A two-sample one-sided paired t-test shows that the improvement in accuracy is significant when using posture and key adaptation versus key adaptation only (t(29) = -2.4421, p = 0.01).

There can be different levels of complexity for posture adaptation. For the most complex one, we can have two models for each key for each input posture, i.e., $N(\underline{\mu}_{y,k}, \Lambda_{y,k})$ for $y \in \{\text{one-finger, two-thumb}\}$; or we can do posture adaptation only for a certain number of keys, for examples for keys on the left side only or a smaller subset of keys on the left side. We experimented with these options and the best result obtained is when posture adaptation is applied to the keys on the left side while keeping the Gaussian models for the keys on the right side independent of postures, i.e., $N(\underline{\mu}_k, \Lambda_k)$. The error rate for posture and key adaption shown in Table 1 is based on this option. This approach is similar to the "parameter tying" technique that is often used in acoustic modeling in speech recognition [4] and natural language processing [14] to deal with insufficient training data.

From (5), the most likely intended key based on posture- and key-adaptive model can be written as

$$\hat{k}_i = \arg\max_k \sum_y p(\underline{c}_i|k, y; \underline{\theta}_s)[[y = \mathsf{y}]], \qquad (10)$$

where

$$p(\underline{c}_i|k, y; \underline{\theta}_s) = \begin{cases} N(\underline{c}_i - \underline{o}_k; \underline{\mu}_{y,k}, \Lambda_{y,k}) & \text{if } k \text{ is on left side} \\ N(\underline{c}_i - \underline{o}_k; \underline{\mu}_k, \Lambda_k) & \text{if } k \text{ is on right side} \end{cases} \qquad (11)$$

The choice of this set of keys is not arbitrary. As observed by Azenkot et al. [2], the difference in horizontal offsets of the touch points from different postures are most prominent for the keys on the left side. In addition, the analysis of variance based on the tapping coordinates from the Pepper dataset shows that, for different postures, there are significant differences in the means of the x coordinate for the keys on the left side of the keyboard ($p < 0.05$).

Figures 3(b) and 3(c) show the comparison of effective areas of the keys with spatial models adapted to one-finger input and two-thumb input respectively. Note how the key areas for the left-side keys shift to the left for two-thumb input, and shift to the right for one-finger input. The difference in the effective key areas is the same concept as key-target resizing mentioned in [8, 18].

The error rate in Table 1 is based on perfect knowledge of the posture. In a deployed system, the online posture classification may introduce errors. To mitigate the problem, the posture-adaptive model can be enabled only if the posture classification confidence is high. The details are explained in "Evaluation of Standalone SBM section.

User Adaptation

The user- and key-adaptive model reduces the error rate by 14.7% over the key-adaptive model. As some data is needed to build the user and key-specific model, the most likely intended key $\hat{k}_i$ can be written as:

$$\hat{k}_i = \arg\max_k \sum_u p(\underline{c}_i|k, u; \underline{\theta}_s)[[u = \mathsf{u}]], \qquad (12)$$

where

$$p(\underline{c}_i|k, u; \underline{\theta}_s) = \begin{cases} N(\underline{c}_i - \underline{o}_k; \underline{\mu}_k, \Lambda_k) & \text{if } i \le T_{\text{user}} \\ N(\underline{c}_i - \underline{o}_k; \underline{\mu}_{u,k}, \Lambda_{u,k}) & \text{if } i > T_{\text{user}} \end{cases} \qquad (13)$$

T_{user} is the minimum number of data points needed to build a user-specific key model. Equation (13) shows the backoff mechanism for the user- and key-adaptive model. The threshold can also be key dependent, i.e. $T_{\text{user, k}}$ for each key k. We have not experimented with this variation and this can be part of the future work.

The result in the last row of Table 1 is computed by the following method. For each fold of the cross-validation, data from the training set are used to train the combined backoff key models. For each user u in the test set, 50% of the data for each key is used to train the user- and key-adaptive model,

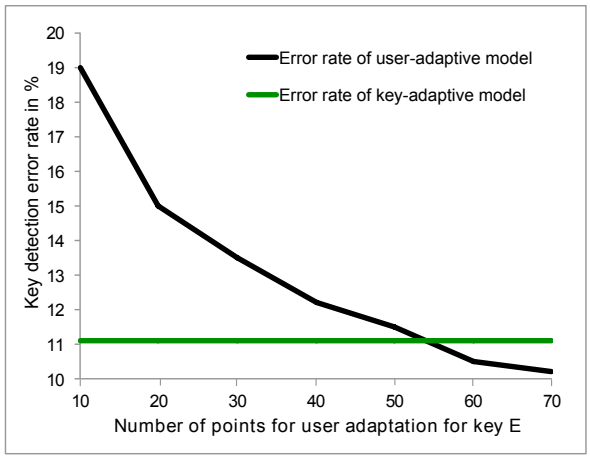


Figure 4. Graph showing how key estimation error rate for key E changes as the number of points used to build the user-adaptive model for key E increases.

i.e., $N(\underline{\mu}_{u,k}, \Lambda_{u,k})$. The intended keys of these data points are evaluated on the combined backoff key models. The remaining 50% of the test data for each user are evaluated on the user- and key-adaptive model.

The black line in Figure 4 shows how key estimation error rate for the "E" key changes as the number of points used to build the personalized model for key "E" increases. The green line indicates the error rate when using the key-adaptive model only. The error rates are obtained using cross-validation and are based on the touch points that are not used in building the model. We choose "E" as an example because it has the most number of data points (around 90) for each user in the Pepper dataset besides the "Space" key. It is hard to do the analysis for the other keys with relatively small number of data points. However, we believe the result would be applicable to other keys. Also due to the limited number of data points, we only show the trend until the number of touch points used to build the user model is 70. Nevertheless, the figure still shows a general trend, and suggests that the minimum number of data points for building a user- and key-adaptive model should be at least 55 when the error rate becomes lower than that when using key-adaptive model alone. This is only a single data point for one key, but since it's cross-validated and averaged over several users, it is probably not too far from the real threshold.

INPUT HAND POSTURE CLASSIFICATION

The posture-adaptive model requires a real-time posture classifier. For this paper, we have developed a classifier that continuously estimates $y \in \{\text{two-thumb, one-finger}\}$ from only the user's input touch points. Note that a variety of sensor, signals, and algorithms could be used, but optimal posture classification is not the primary goal of this paper.

Our analysis of the touch point data shows that for one-finger input, the time taken increases with the distance moved, whereas for two-thumb input, there is no obvious trend. The difference is more significant when the log distance (natural log) between two consecutive touch points is greater than 5. The result makes intuitive sense. For example, a user may take a longer time than average to type the letters "AL" using one finger because the finger has to travel a long distance from the "A" key to the "L" key on a Qwerty keyboard. But with two thumbs, typing "AL" can be faster than average because different hands are used for each key. No long distance move is required.

Based on this finding, we include the time elapsed and the log distance between two consecutive touch points as two features for the posture classifier. To account for individual typing speed differences, we also use normalized time elapsed between consecutive key presses as the third feature. It is calculated by dividing the time elapsed since the last touch point by the average time elapsed for the last 10 touch points.

Using 20 users data for training and 10 users data for testing, we train an SVM classifier offline using these three features. Each input touch point satisfies condition C: it is on the other side of the keyboard from the previous touch point, and the log distance from the previous touch point is at least 5. This classifier gives a probability score p_y^{single} for these touch points. Note that $\sum_{y \in \mathcal{Y}} p_y^{\text{single}} = 1$. The posture classification accuracy for these touch points is 83.6%.

In order to classify every key tap and assuming the user does not change posture rapidly, we look at a sliding time window of 10 touch points (about 2 words). For each time window, we use another SVM classifier with the following features:

1. Correlation between time elapsed and log distance (this feature has the advantage of being speed independent)
2. Sum of $p^{\text{single}}_{\text{one-finger}}$ for touch points satisfying condition C.
3. Sum $p^{\text{single}}_{\text{two-thumb}}$ for touch points satisfying condition C.
4. Number of touch points within the window classified as one-finger input.
5. Number of touch points within the window classified as two-thumb input.

Features 2-4 are also normalized by the window size. The history of the touch points are cleared for every new typing session. The choice of the size of the sliding window represents a trade-off between the accuracy of the classification and how responsive the system is when the user changes posture. We assume that in general users do not change posture more often than every two words.

The sliding time window classifier gives a final probability score p_y for posture y for each touch point. Again $\sum_{y \in \mathcal{Y}} p_y = 1$. To evaluate the classification accuracy, we set the the classified posture to be y if $p_y > 0.5$. With this threshold, the overall classification accuracy for each touch point with a sliding time window is 86.4% (23,128 out of 26,769 touch points).

In the sliding window approach, the posture for the first few touch points of a new session is unknown. In this situation,

Highest level spatial model in SBM	Character error rate (SD)
Base model	8.64% (5.1)
Key-adaptive model	8.71% (5.1)
Posture- and key-adaptive model	8.39% (5.0)
User- and key-adaptive model	7.62% (4.8)
Posture-, user- and key-adaptive model	7.50% (4.5)

Table 2. Comparison of character error rates using the real-world implementation of the standalone SBM model across different submodels. These results are based on *inferred* postures and intended keys for building the user specific models. Results are based on 10 test users.

a system can back off to a lower level spatial model (key-adaptive or base model). Furthermore it can enable the posture adaptive spatial model only when the probability score for one posture is much higher than the other.

IMPLEMENTATION AND EVALUATION

The key estimation process with the proposed SBM fits nicely with the Chain of Responsibility design pattern. Each higher level model can hold one or more references to lower level models. Given a touch point, the system queries the highest level model for a Gaussian submodel for a particular user/posture/key combination. If not present, the higher model calls the lower model, and the query propagates until a Gaussian submodel is found.

Based on this design, we implemented a prototype of SBM with online posture classification. The prototype has flags to turn on/off certain adaptive models so that we can easily compare their performance. For all the evaluations, we use 20 users' data for training both the posture classifier and the spatial models, and the other 10 users' data for testing.

Evaluation of Standalone SBM

Like Findlater et al. [6] and Rudchenko et al. [18], we evaluate the effectiveness of the standalone SBM prototype on key estimation without a language model. Table 2 and Figure 5 show the comparison of character error rates using SBM with different submodels turned on. The main difference between these results and those in Table 1 is that in the prototype implementation, posture is *inferred* and the user specific model is built according to *inferred* user-intended key. This means that the true identity of the intended keys are unknown which is true in a realtime application. The implication is that some of the data points used to train the posture- and/or user-specific models are inevitably erroneous. Our goal is to reduce the error as much as possible and to show that even with potential errors, the overall error rates with adaptive models are still lower than the non-adaptive one.

Posture Adaptation

The posture- and key-adaptive model uses the posture classification method described in the previous section. As a result, the key detection error rate for this model is compounded by the posture classification error rate.

An error in posture classification will lead to an incorrect submodel, and hence adversely affect the key detection accuracy. To minimize incorrect classifications, we consider

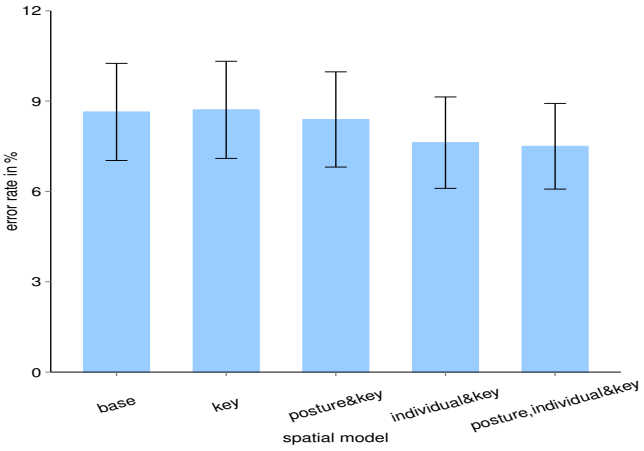


Figure 5. Bar plot of error rates in Table 2 with error bars indicating standard error of the mean (SEM).

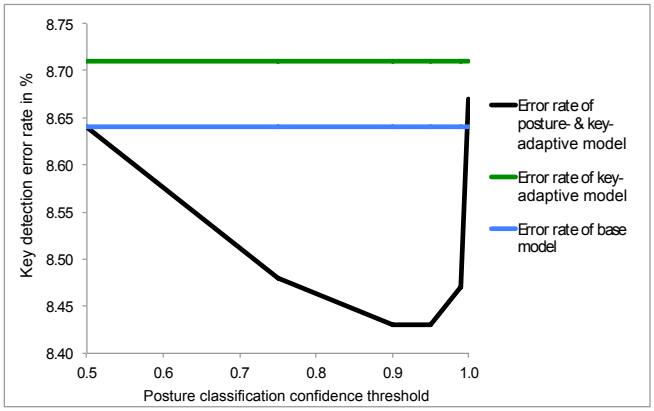


Figure 6. Graph showing how key estimation error rate changes as the confidence threshold for posture classification increases.

the confidence of the classifier. The classifier's confidence can be determined by examining the pair of probability scores $(p_{\text{one-finger}}, p_{\text{two-thumb}})$. We can set a threshold T_{posture} such that the input posture is classified as y only if $P_{\mathsf{y}} \geq T_{\text{posture}}$. Otherwise the posture is treated as unknown and we backoff to a lower-level spatial model.

There is a trade-off in setting the T_{posture} threshold. When T_{posture} is higher, there will be fewer errors in posture classification, but also more touch points classified as unknown posture. No posture adaptation can be used for these touch points. Figure 6 shows that there is an optimal level of the threshold beyond which the error rate increases because we can no longer take advantage of posture adaptation. The error rate in Table 2 for SBM with posture- and key-adaptive model as the highest order model is obtained by setting $T_{\text{posture}} = 0.94$.

User Adaptation

When computing the user- and key-adaptive model, for every touch point we calculate the probability for each key given the underlying spatial model. Then we use the (x, y) coordinates of the touch point to update the Gaussian model for the most probable key. Updating the Gaussian model involves comput-

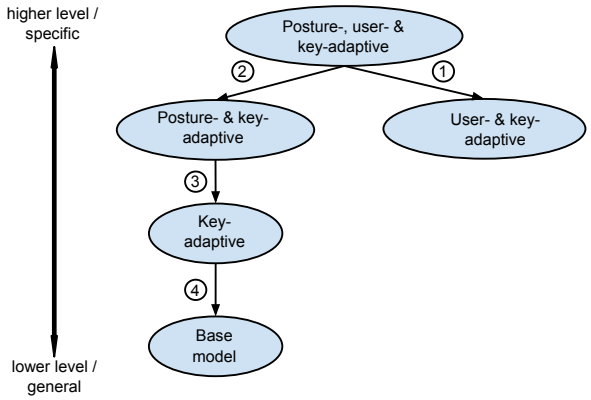


Figure 7. Partial hierarchical model implemented in our prototype. The numbers indicate the order of the submodels to use during backoff.

ing the running average of the (x, y) offsets from the center of the key and the covariance matrices. The counter for the number of points used for a particular key Gaussian is maintained so that we know when a particular Gaussian model is *mature*. When there are not enough data points, the system backs-off to a lower level model.

We set the minimum number of points needed to build the Gaussian model for a particular key and user pair to be 50. Ideally we would want to set this number higher for better model reliability as shown in Figure 4. However the current number is limited by the amount of data available. Only about half of the keys have more than 50 data points for each user. The average number of data points per key per individual is about 80. We chose 50 as a middle ground so that there is room for individual adaptations for most of the keys. In a deployed system, the data limitation will be less of an issue since the user is continually typing. Alternatively, the system can use a user's past data to update the combined model.

Posture, User, and Key Adaptation

Tying everything together, the last row in Table 2 shows the error rate when the highest level model is posture-, user- and key-adaptive. It reduces the error rate of the base model by 13.2% $((8.64 - 7.50)/8.64)$. The reduction is significant based on the two-sample one-sided paired t-test (t(9) = 2.58, p = 0.015).

Figure 7 shows the backoff mechanism when there is insufficient data for the highest level model. The results based on the Pepper dataset in the "Comparison of Spatial Models" section suggest that we can give higher priority to the user- and key-adaptive model when there is not enough data for the highest level model. If there are still not enough data for that, we can further backoff to the posture- and key-adaptive model.

Note that because the dataset is a between-subjects study, we cannot study the full effect of posture adaptation for each individual. We believe that there could be additional improvement, but further study is required to verify it.

DISCUSSION AND FUTURE WORK

We have shown the potential of posture, user, and key adaptation for improving key estimation. An important limitation of our exploration thus far has been the constraints of the Pepper dataset. To advance this line of work further we need more data than is typically available from lab experiments. Methods such as real-use logging and game playing [18] can be employed to gather a large body of data. With this scale of data, the specific (sub)models, the backoff procedure and sequence, and the parameters learned from the data may change and be better optimized in the future.

In our proposed SBM, we switch from a more general to a more specific submodel when a certain threshold of confidence is reached or enough data points are collected. This is a binary decision, i.e., a submodel is either used or it is not. Alternatively, we can employ a mixture model that is a weighted average of the general and the specific submodels (i.e., using soft decisions), with the relative weight of the specific submodel increasing as confidence in it increases. We have started to experiment with this approach for user adaptation, i.e., combining the individual data points with the combined model from the very beginning instead of waiting until enough individual data points have been collected. Our preliminary studies have not shown a substantial difference, but our ongoing investigations may yield better results. It is also worth noting that even though the mixture model approach may not improve accuracy, it may provide a more gradual transition between submodels and hence may provide a less disruptive user experience.

Once more data are collected to improve the SBM, more user studies can be run to validate how the system improves user typing accuracy and speed. Of particular interest is how users adapt their behavior to the SBM. For example, since posture adaptation can tolerate different shifts left or right, might users respond by shifting their touch positions even further? Will users adopt the tactic of choosing one posture and sticking to it in order to increase the system's accuracy, and would it be desirable for them to do so? Might the sudden switch by the system from one model to another disrupt interaction by making the input method less predictable? All these are interesting HCI questions to be explored.

In our analysis, we only considered right-handed users. For left-handed users, we expect the posture-adaptive models would be different, for example, the horizontal offsets would be in the opposite direction. This means we also need a mechanism to identify the handedness of the user which is also an interesting aspect of the future work.

In this work, we combined the index-finger and one-thumb data together because with the current touchscreen phones it is difficult to reliably distinguish the two postures. But it may be possible to identify them by augmenting the capacitive sensing with a range of frequencies that can help identify different kinds of hand grips [19]. Another possibility is to add a widget to the keyboard that allows the user to optionally specify what posture they are currently using. While this method can eliminate the posture prediction error, it does so at the expense of overhead for the user. It would be interest-

ing to examine the trade-off between accuracy and time, and users' preferences in this regard.

We have so far only considered the spatial model alone. More work is needed to explore the interaction between spatial adaptation and language modeling. As the language model can get fairly complex, our plan is to start with the simplest one, e.g., one based on dictionary word frequencies, and then add complexity step by step by, e.g., considering bigram frequencies, substitution, elimination, and transposition errors.

Additional improvements in key estimation may be possible by extending the formulae in the "Key Estimation Formulation" section to consider a sequence of touch points [7, 8] instead of just one. It would also be interesting to study how to use posture detection to guide transposition error correction, because transposition errors are more likely to occur when two hands are used to type.

CONCLUSION

We have introduced and evaluated a novel hierarchical adaptive spatial model for touchscreen keyboards. Through comparative submodel analysis and evaluation of a prototype implementation, we have shown that both posture and user adaptation for a spatial model can improve key estimation accuracy. When posture, user, and key adaptations are combined, they achieve the greatest improvement. For real-world implementations, the hierarchical structure gives a systematic way of backing-off to successively simpler models when data is limited or when there is insufficient confidence in higher level models.

We have also developed a new touchscreen input posture classification method that achieves an accuracy of 86.4% for classifying one-finger and two-thumb input. When combined with the adaptive spatial model, the our analysis shows that the overall key detection increases significantly.

Our work also opens up many more interesting HCI questions in adaptive STK that may not have been previously considered. We think that addressing these questions will help improve the user experience with STK as touchscreen devices become even more ubiquitous.

ACKNOWLEDGEMENTS

We want to thank our team members Ciprian Chelba for sharing his theoretical insights and for extensively commenting an early draft of this paper, Xiaojun Bi for sharing tools for collecting and analyzing the data, and Shiri Azenkot for collecting the Pepper dataset [2] reused in the analysis of SBM methods proposed in this paper.

REFERENCES

1. Al Faraj, K., Mojahid, M., and Vigouroux, N. Bigkey: A virtual keyboard for mobile devices. In *Human-Computer Interaction. Ambient, Ubiquitous and Intelligent Interaction*, vol. 5612. Springer Berlin / Heidelberg, 2009, 3–10.
2. Azenkot, S., and Zhai, S. Touch behavior with different postures on soft smart phone keyboards. In *Proc. MobileHCI* (2012).
3. Bao, P., Pierce, J., Whittaker, S., and Zhai, S. Smart phone use by non-mobile business users. In *Proc. MobileHCI '11* (2011), 445–454.
4. Bellegarda, J., and Nahamoo, D. Tied mixture continuous parameter models for large vocabulary isolated speech recognition. In *International Conference on Acoustics, Speech, and Signal Processing*, vol. 1 (may 1989), 13 –16.
5. Brewster, S., Chohan, F., and Brown, L. Tactile feedback for mobile interactions. In *Proc. of CHI '07* (2007), 159–162.
6. Findlater, L., and Wobbrock, J. Personalized input: improving ten-finger touchscreen typing through automatic adaptation. In *Proc. CHI '12* (2012), 815–824.
7. Goodman, J., Venolia, G., Steury, K., and Parker, C. Language modeling for soft keyboards. In *Proc. IUI '02* (2002), 194–195.
8. Gunawardana, A., Paek, T., and Meek, C. Usability guided key-target resizing for soft keyboards. In *Proc. IUI '10* (2010), 111–118.
9. Hoggan, E., Brewster, S. A., and Johnston, J. Investigating the effectiveness of tactile feedback for mobile touchscreens. In *Proc. of CHI '08* (2008), 1573–1582.
10. Jameson, A., Großmann-Hutter, B., March, L., and Rummer, R. Creating an empirical basis for adaptation decisions. In *IUI 2000: International Conference on Intelligent User Interfaces*, H. Lieberman, Ed. ACM, 2000, 149–156.
11. Jameson, A., Schäfer, R., Weis, T., Berthold, A., and Weyrath, T. Making systems sensitive to the user's time and working memory constraints. In *Proceedings of the 4th international conference on Intelligent user interfaces*, IUI '99 (1999), 79–86.
12. Katz, S. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *Acoustics, Speech and Signal Processing, IEEE Transactions on 35*, 3 (mar 1987), 400 – 401.
13. Kristensson, P., and Zhai, S. Relaxing stylus typing precision by geometric pattern matching. In *Proc. IUI '05* (2005), 151–158.
14. Lin, Y.-C., Chiang, T.-H., and Su, K.-Y. The effects of learning, parameter tying and model refinement for improving probabilistic tagging. *Computer Speech and Language 9*, 1 (1995), 37 – 61.
15. MacKenzie, I. S., and Zhang, S. X. The design and evaluation of a high-performance soft keyboard. In *Proc. CHI '99* (1999), 25–31.
16. Magnien, L., Bouraoui, J. L., and Vigouroux, N. Mobile text input with soft keyboards: Optimization by means of visual clues. In *MobileHCI '04*, vol. 3160. Springer Berlin / Heidelberg, 2004, 197–218.
17. Rashid, D. R., and Smith, N. A. Relative keyboard input system. In *Proc. IUI '08* (2008), 397–400.
18. Rudchenko, D., Paek, T., and Badger, E. Text text revolution: a game that improves text entry on mobile touchscreen keyboards. In *Proc. the 9th international conference on Pervasive computing* (2011), 206–213.
19. Sato, M., Poupyrev, I., and Harrison, C. Touché: Enhancing touch interaction on humans, screens, liquids, and everyday objects. In *Proc. CHI '12* (2012).
20. Zhai, S., Hunter, M., and Smith, B. A. The metropolis keyboard - an exploration of quantitative techniques for virtual keyboard design. In *Pro. UIST '00* (2000), 119–128.
21. Zhai, S., Sue, A., and Accot, J. Movement model, hits distribution and learning in virtual keyboarding. In *Proc. of CHI '02* (2002), 17–24.
22. Zitouni, I. Backoff hierarchical class n-gram language models: effectiveness to model unseen events in speech recognition. *Computer Speech and Language 21*, 1 (2007), 88 – 104.