Automated Smartphone Keyboard Error Corrections

Vered Aharonson[1,2(✉)], Rotem Rousseau[3], and Eran Aharonson[3]

[1] Department of Electrical Engineering,
Afeka Tel Aviv Academic College of Engineering, Tel Aviv, Israel
[2] School of Electrical and Information Engineering,
University of the Witwatersrand, Johannesburg, South Africa
vered.aharonson@wits.ac.za
[3] Department of Software Engineering,
Afeka Tel Aviv Academic College of Engineering, Tel Aviv, Israel
erana@afeka.ac.il

Abstract. The usability of smartphone's touch keyboard is often hampered by typing mistakes resulting from the small size of the virtual keys relatively to the user's finger size. Although this problem has been addressed in various methods, an optimal solution in terms of both accuracy and user experience, however, has not been achieved. We developed an algorithm that predicts users typing intentions based on a statistical geometrical modeling of the touch points area. The algorithm builds a user-adaptive virtual location of the key based on deviations probability computation. An uncertainly measure activates a language statistics engine to enhance the prediction. The algorithm was integrated into the default Android® keyboard and was tested on users. Typing error rate using the implemented algorithms was reduced by 23.1% on average. The proposed method can enhance typing accuracy and user experience and may facilitate and improve the design of smaller and cheaper touch based smartphones.

Keywords: Smart keyboard · Enhanced typing experiences
Typing error corrections · Location statistics

1 Introduction

A vast majority of smartphones applications use text inputs and entail typing. The user experience when typing on a smartphones' virtual keyboard is often hampered by typing errors. One reason for these errors is the small size of the keys relatively to the user's finger [1, 2]. Another may be shifts in the user's finger placement due to eye – finger coordination [3–5].

A prevalent solution to this problem, which is implemented in many smartphones is a language-based auto-correction of the errors [6], which predicts the letter typed according to its highest probability in the word or sentence (Fig. 1).

Many users, however, find this solution irritating, since the predicted word often does not match the user's intention, especially when it comes to non-vocabulary words like names and abbreviations, or to uncommon languages [7]. Although this technique

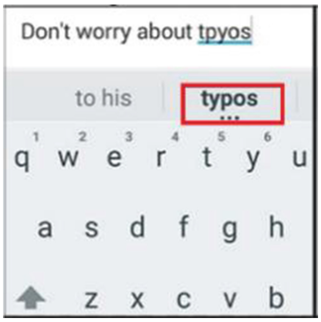

Fig. 1. Illustration of a keyboard language based auto-correction

improved considerably over the years, especially where a user's language adaptation was implemented, (i.e. SwiftKey®, Google® keyboard, Fleksy®, Swype®, Minuum®) these limitations have not yet been fully resolved.

Recent studies explored the association between the geometrical properties of a user's touch points and the intended key. Azenkot et al. [4] investigated the average deviations between the positions of each key to the users' touch points. Their study illustrated various patterns in the offsets between the different touchpoints and their intended keys. The authors proposed a method called `Remulation' for real time corrections based on their users' studies [8]. The method is based, however, on prior user's typing patterns database and is not personalized to a specific user.

The goal of our study is to develop and implement an algorithm that can provide typing accuracy while maintaining the user experience. The algorithm is user adaptive and does not necessitate training.

2 Methods

Our design entails an adaptive keyboard that adjusts for key offsets in the individual user typing. The keyboard learns the association between the user's touch locations and the intended key. A hybrid configuration design combines this geometrical analysis of the user's touch point and a language analysis of the typed keys. An unsupervised learning design does not require prior knowledge or data base.

2.1 Algorithm

An The algorithm produces an "intended key" estimation for each location of the user's touch, denoted Touchpoint. The computation is based on a first order Markov chain model, updating for each additional touchpoint. The initial state, or TouchPoint, of the chain is the center of the keyboard key. A geometrical location estimation, denoted TouchMap, is computed as the weighted average of the current and last TouchPoints for each key (Eq. 1). The weights are heuristically chosen, based on the observation that the current touch points have more strength than past ones.

$$TouchMap = 0.9(TouchPoint_{current}) + 0.1(TouchPoint_{last}) \quad (1)$$

The association of every new TouchPoint, with an "intended key" is based on the TouchPoint's distance from all TouchMaps:

The three smallest distances are selected as 'candidates'. If one of these 3 values is smaller by more than 5% from the other 2, it is selected as the "intended key". If two of the three or all three values are less than 5% apart from each other in terms of distance, the decision is of uncertainty and a second phase of the algorithm is activated

The second phase is based on language prediction: each of the "intended key" candidates is tested using its bigram and trigram with the previous letters in the typed word and the probability of these n-grams is retrieved from a database dictionary. The highest probability ranks the chosen candidate. If two candidate letters have identical probability, one of them is randomly chosen.

A flow diagram of the algorithm is presented in Fig. 2.

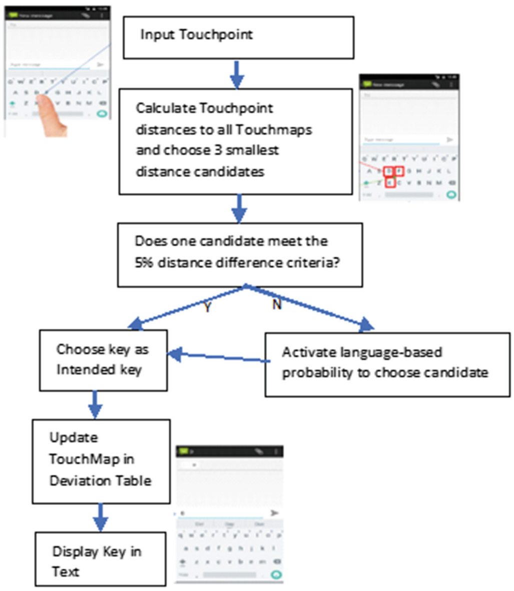
Input Touchpoint
Calculate Touchpoint distances to all Touchmaps and choose 3 smallest distance candidates
Does one candidate meet the 5% distance difference criteria?
Y
N
Choose key as Intended key
Activate language-based probability to choose candidate
Update TouchMap in Deviation Table
Display Key in Text

Fig. 2. Algorithm flow chart

2.2 Implementation

Signal The system includes three main components: a user interface, which is the original user interface of the smartphone's default keyboard, the algorithm's implementation and a database for later analysis and testing.

The implementation is designed to fit Android smartphones from version 4.1 or higher, which can accommodate the majority of Android devices available today.

The algorithm was implemented using Android Studio® and Google®'s open source default keyboard software.

The user interface is identical to the standard Smartphone keyboard except of an optional additional key which activate a new debug visual tool called TouchpointView. The tool allows a qualitative examination of the TouchPoints. As illustrated in Fig. 3, the additional key displays a circle and an "x" which lays over the keyboard a display of the actual touch points of the user's typing and clears them, respectively.

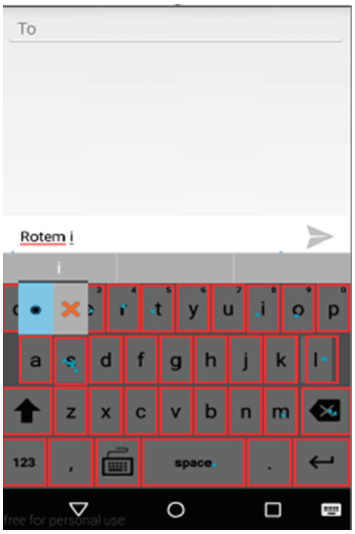

Fig. 3. TouchpointView tool display of the touch points (small blue dots) upon pressing the additional button (circle and "x").

The TouchpointView tool is also implemented as an option in the user interface, for users who wish to observe their own key touch distributions.

2.3 Performance Evaluation

Ten users participated in a performance evaluation experiment. The users were asked to type a set of 21 short sentences on an Android smartphone, using the default smartphone keyboard. The sentences contained 86 words and 405 letters which were uniformly distributed among the 26 English letters. The users did not see the result of their typing, except from an asterisk symbol indication that a letter was typed. The same experiment was performed twice, on two different smartphones, with different screen sizes. The Smartphones used were Galaxy S5 – as a large screen example (5.1 in.) and Posh Micro X SN40 – the smallest Android smartphone (2.4 in.). The keyboard key sizes are 182 × 108 pixels and 52 × 24 pixels. For the large and small screens, respectively.

The performance, in terms of error rate, was compared between the default keyboard and when running the algorithm. In order to evaluate the contribution of the language based key prediction to the basic geometrical model, the performance of the two versions of the algorithm: with and without the language layer were compared, for the two screen sizes. The performance measure was error rate: ratio of key errors to the number of TouchPoints, in percentage.

Error rates were compared between the two keyboard implementations: standard and enhanced, and between the two screen sizes.

3 Results

The Fig. 4 presents the error rates of the 10 subjects when typing on the two screen sizes. Both the standard keyboard's results and the ones acquired when implementing our algorithm ("enhanced" are presented for both screen's experiments.

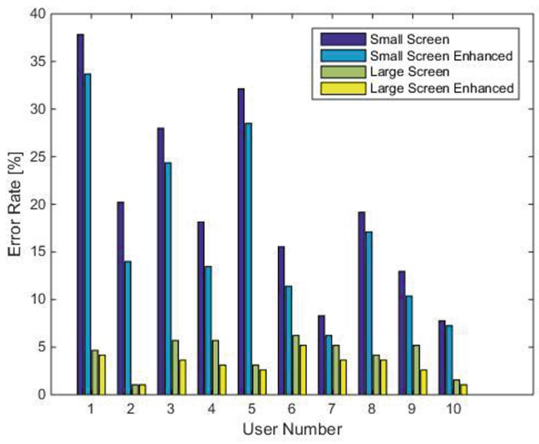

Fig. 4. Bar graph of the users' error rate for the two screen sizes and for the standard and enhanced keyboards implementations.

The error rates' means and standard deviations (SD) for the two keyboard implementations, standard and enhanced, and for the two screen sizes are presented in Table 1.

Table 1. Average users error rates.

Error rate [%] mean ± SD	Standard keyboard	Enhanced keyboard
Small screen	20.0 ± 9.9	16.63 ± 9.2
Large screen	4.2 ± 1.6	3.0 ± 1.1

It is obvious that the smaller screen induces significantly more errors. The results indicate that error rate was reduced when the enhanced keyboard was implemented, for both screen sizes. The reductions in error rate were 18.1% and 22.9% for the small and large keyboard, respectively.

A further analyzes examined which of the correction layers: geometric or language based contributed to the error correction. The results are presented in Table 2.

Table 2. Relative contribution to error rate corrections.

Error rate correction improvement [%]	Geometric correction algorithm	Language-based correction algorithm
Small screen	29.38	70.62
Large screen	77.50	22.50

The results indicate that the geometric correction is able to attend to more than 75% of the typing errors for the large screen, whereas in the small screen case, most errors necessitated an activation of the language-based correction.

4 Discussion and Conclusions

We presented an algorithm that could correct typing errors on touchscreen, primarily by examining the geometrical distribution of a user's TouchPoints and determining the user's intended key according to distances minimization. The algorithm is implemented in a hybrid structure, such that in case of uncertainty in the geometrical decision, a language-based decision is activated, which takes into account the letters bi-grams probabilities. The algorithm is continuously updating its decision with each TouchPoint employing an unsupervised decision logic, and therefore does not require any training by the user.

The experiment was performed on two screen sizes and indicated that the algorithm was able to substantially reduce the error rates in both screens.

When the contribution of each part of the hybrid system; geometric and linguistic on the performance was examined, a linguistic error correction was needed for 70% of the error corrections in the small screen as compared to 22.5% of the cases in the large screen experiments. As could be predicted, the uncertainly in the geometric decision is larger for smaller keys and hence a language based decision is required in most cases for the smaller screens.

The results imply that this enhanced keyboard is more accurate and more user-adaptive than the standard keyboard of Android smartphones available in the market today. Its implementation is transparent to the user and does not necessitate any user interface changes.

Healthy subjects' typing errors are not large in size and their rate, especially on large keyboards, is relatively small. A further study will examine if our system could be augmented to accommodate people with both vision and motor disorders or disabilities [9].

References

1. Sears, A., Revis, D., Swatski, J., Crittenden, R., Shneiderman, B.: Investigating touchscreen typing: the effect of keyboard size on typing speed. Behaviour & Information Technology **12** (1), 17–22 (1993)
2. Findlater, L., Wobbrock, J. O., Wigdor, D.: Typing on flat glass: examining ten-finger expert typing patterns on touch surfaces, pp. 2453–2462
3. Kwon, S., Lee, D., Chung, M.K.: Effect of key size and activation area on the performance of a regional error correction method in a touch-screen QWERTY keyboard. Int. J. Ind. Ergon. **39**(5), 888–893 (2009)
4. Azenkot, S., Zhai, S.: Touch behavior with different postures on soft smartphone keyboards, pp. 251–260
5. Nicolau, H., Jorge, J.: Touch typing using thumbs: understanding the effect of mobility and hand posture, pp. 2683–2686
6. Kukich, K.: Techniques for automatically correcting words in text. ACM Computing Surveys (CSUR) **24**(4), 377–439 (1992)
7. Tetariy, E., Bar-Yosef, Y., Silber-Varod, V., Gishri, M., Alon-Lavi, R., Aharonson, V., Opher, I., Moyal, A.: Cross-language phoneme mapping for phonetic search keyword spotting in continuous speech of under-resourced languages. Artificial Intelligence Research **4**(2), p72 (2015)
8. Bi, X., Azenkot, S., Partridge, K., Zhai, S.: Octopus: evaluating touchscreen keyboard correction and recognition algorithms via, pp. 543–552
9. Trewin, S., Pain, H.: Keyboard and mouse errors due to motor disabilities. Int. J. Hum Comput Stud. **50**(2), 109–144 (1999)