Automatic Flick Keyboard Error Correction Based on GMM-LR-LM

Gang Qiao, Yihua Huang, Yabo Li, Jianmin Wu, Tianhuang Su

EBG, Baidu Inc, China, 518052

{qiaogang01, huangyihua, liyabo01, wujianmin01, sutianhuang}@baidu.com

1.Introduction

With the rapid development of mobile Internet, Social apps is becoming more and more important. As an indispensable app of intelligent equipment, the input method is very important and because of the limited size of smartphones, the auto-correction function has been paid more and more attention by users.

Currently the most popular Japanese input method in touch screen smartphone is flick keyboard with kana-kanji convert. As showed in Figure 1, kanas and functional keys are assigned to 12 keys. Each key expresses 3~5 kanas that share generally the same consonant corresponding to one line in the gojūon order. A user is supposed to enter a kana with vowel "a" by tapping the key face, and rest kanas of the same gojūon line by swiping to certain direction for sub-keys. Although flick keyboard has bigger key face size than QWERTY keyboard, the combination of tapping and swiping makes errors happen more frequently. The probability of kana error is near to 10% per word according to volunteers' feedbacks，most of which are substitution errors. The substitution could be any combination of neighboring keys and are too hard to be enumerated by hand-written rules.

Figure 1: The most popular Japanese Input Method

Auto-correction [1] has been proposed very early, with the purpose to find the users' errors and correct them. Edit distance [2] is still widely used today. This algorithm is simple and easy to understand. For example, if you type "あ" but truly to have "か", the substitution will analyze the probability $p(か|あ)$ which can be trained from corpus. But the performance is not very good because of few features. [3] adopted machine learning in spelling correction, but it only applies to search engines.

Fortunately, we found that most of the substitution errors fall into two categories. One is by tapping on or swiping to a wrong key, and the other is by taking a tapping operation as a swiping one or vice versa. Therefore, the auto-correction problem consists of two parts. Firstly, given a sequence of touch operations that may contain substitution error, the most likely button sequence should be identified. Secondly, a series of kana sequences with probability should be generated based on the button sequence. To address this problem, we exploit the characteristics of touch screen interaction and propose a probabilistic model for flick keyboard auto-correction.

2. Methods

The GMM-LR-LM model is proposed to solve this problem. GMM (Gaussian Mixture Model) is used to map the input touch sequence to the button sequence set. LR (Logistic Regression) is used to map the button sequence to the kana sequence and the LM (Language Model such as N-gram) is used to adjust the final probability of kana sequence.

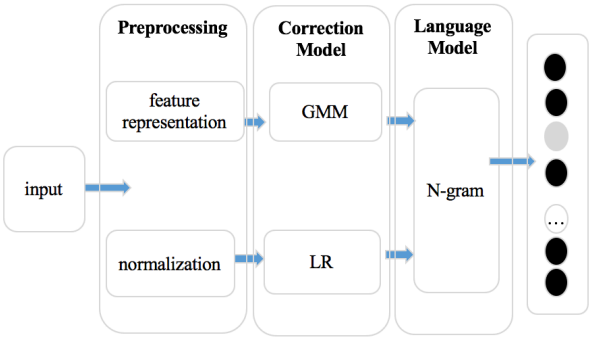

Figure 2: The framework of the proposed method

The paper framework is showed in Figure 2, which can be divided into three parts: Preprocessing, Correction Model and LM. In the first part, the raw touch sequence data is extracted as a feature vector and normalized to a standard one. The correction work in done in the second part and the third part deals with LM.

2.1. Preprocessing

2.1.1 Feature representation

In order to predict kana exactly, features are collected such as touch point coordinate (x, y) and touch duration. Different feature combinations are used in different models. Generally, a touch point can be represented as,

$$v = (x, y, duration) \quad (1)$$

The representation of a flick track only consider a start point v_s and an end point v_e, which is shown as,

$$e = (v_s, v_e) \quad (2)$$

2.1.2 Normalization

As we know, smartphone screen resolution varies and the inputs scale differently. But they can be normalized to a standard one according to the keyboard width and height.

$$(x, y) = \left(\frac{x_{real}}{width}, \frac{y_{real}}{height}\right) \quad (3)$$

where (x_{real}, y_{real}) is the real data coordinate; $(\mathrm{width}, \mathrm{height})$ are the keyboard width and keyboard height.

Different features have different units. In order to make the model training more efficient, continuous features are normalized by the formula below, where u, σ are the mean and standard deviation of feature f.

$$f' = \frac{f - u}{\sigma} \quad (4)$$

2.2 Correction Model

In our correction model GMM-LR, for a given touch sequences $E = e_1 e_2, \ldots, e_m$, the output is to generate a number of kana sequences $C_i = c_1 c_2, \ldots, c_m$ with probability $p(C_i|E)$. We assume that each type activity is independent, then we get,

$$p_{gmmlr}(C_i|E) \approx \prod p_{gmmlr}(c_i|e_i) \quad (5)$$

$$p_{gmmlr}(c_i|e_i) = p_{gmm}(b_i|e_i) p_{lr}(c_i|e_i, b_i) \quad (6)$$

where, $p_{gmm}(b_i|e_i)$ is the probability of button b_i and $p_{lr}(c_i|e_i, b_i)$ is the conditional probability of kana c_i based on b_i. Both will be discussed in details below.

2.2.1 GMM: Probability of buttons.

Given a start point feature vector v, the probability of button b_i is denoted as $p(b_i|v)$. In flick keyboard, we assume that the touch point coordinates of the button follow a two dimensional Gaussian distribution which proved by the stat of button "あ" from volunteers' feedback, as shown in Figure 3.

Figure 3: The button touch coordinates

Therefore, GMM [4] with K Gaussian is adopted in our model, where K=12 according to the flick keyboard. The probability of vector v can be formulated as below,

$$p(v) = \sum_{i=1}^{K} \phi_i N(v|u_i, \Sigma_i) \quad (7)$$

$$N(v|u_i, \Sigma_i) = \frac{exp\left[-\frac{(v-u_i)^T\Sigma^{-1}(v-u_i)}{2}\right]}{\sqrt{(2\pi)|\Sigma|}} \quad (8)$$

The parameter ϕ_i is the weight of every Gaussian distribution and u_i, Σ_i are the mean and variance of Gaussian distribution. For a new given v, the probability of $p(b_i|v)$ can be formulated as,

$$p(b_i|v) = \frac{\phi_i * N(v|u_i, \Sigma_i)}{\sum_{j=1}^{K} \phi_j * N(v|u_j, \Sigma_j)} \quad (9)$$

2.2.2 LR: conditional probability of kanas

Given a flick track e, it is a key point to determine the input is a tapping or a swiping. The flick track can be represented as a vector in the formula (1).

This problem can be modeled as a binary classification. LR [5] is adopted in the part.

$$p_{lr}(e) = \frac{1}{1 + exp(-w * e)} \quad (10)$$

W is the parameters in LR model. The output $p_{lr}(e)$ can be regard as the swiping probability, and thus typing probability is $(1 - p_{lr}(e))$.

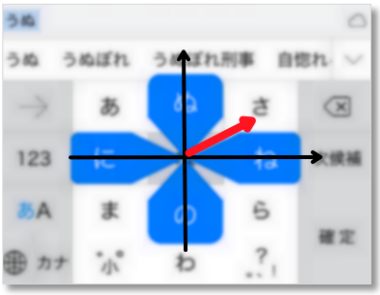

Figure 4: the swiping axis

After we get the probability of swiping, which should be assigned to the four direction kanas. It can be measured by the similarity of vectors. As the Figure 4 shows, the $(-1, 0)$, $(0, 1), (1, 0), (0, -1)$ can be used to represent the left, up, right and down direction vector while $(x_e - x_s, y_e - y_s)$ is the flick direction vector. Then the conditional probability assignation can be formulated as below,

$$p(c_i|e, b) = p_{lr}(e) * \frac{exp^{weight_i}}{\sum_{j=1}^{4} exp^{weight_j}} \quad (11)$$

The weight_i is determined by the angle θ_i between the flick direction and corresponding axis direction.

$$weight_i = \begin{cases} 0 & if\ \theta_i > \frac{\pi}{2} \\ cosine(\theta_i) & else \end{cases} \quad (12)$$

2.3 Language Model

Through the above model, the touch point sequences E are converted to the kanas sequences $C_i = (c_1, c_2, \ldots c_m)$ with probability $p_{gmmlr}(C_i|E)$. But in fact, the kanas itself are not independent. The relations of kanas show in the Figure 5,

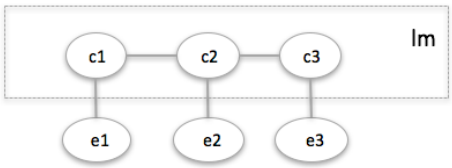

Figure 5: Language model of kanas

So we can build the transition probability model between kanas, and use this model to optimize the probability. A character-level N-gram language model for kana is calculated below,

$$p(C_i) = \prod_{i=1}^{m-n+1} p(c_i|c_{i-n+1}, \ldots, c_{i-1}) \quad (13)$$

The final model GMM-LR-LM can be seen as a linear combination of Language model and Correction model. The probability kana sequence $p(C_i|E)$ is formulated below,

$$p(C_i|E) = \alpha p_{gmmlr}(C_i|E) + (1-\alpha)p_{ngram}(C_i) \quad (14)$$

Finally, we can search the words in dictionary according to the kana sequence. The model will generate more different kana sequences, and every sequence has a proper probability. These probabilities can affect the order of words.

3 Experiment

3.1 Data and Evaluation

The training dataset is constructed from volunteers' feedbacks. An entry consists of the

raw input trajectory and the original button that the keyboard recognizes. 101K entries are used to train the GMM-LR model. Regarding the test dataset, we invited a group of colleagues to input specified sentences in hiragana by conventional flick keyboard. If input error happens, they are asked to keep it. Finally 5350 entries are collected for test dataset，including 510 wrong entries.

The test set of data are mapped to the same keyboard layout input method, according to the auto-correction model of the input method, combined with the language model, calculate the correct entry number of words and wrong correct entry number. In fact, we only pay attention on the first candidate from input method applications. So if one case is good, the first candidate must equal to the target, else this case is bad. The evaluation measure is conventional Recall, Precision and F-score.

3.2 Experiment and Result

In Correction model, the conventional EM algorithm is used to train the GMM model and K is set to 12 in MM. LR learning algorithm is SGD, and the loss function is cross-entropy. N-gram is set to 3.

Table 1: The result of the contrast experiment

	Recall	Precision	F-score
Third-Party	58.8%	93.75%	72.3%
Rule-based	19.6%	90.5%	39.2%
GMM-LR-LM	72.5%	94.8%	82.2%

This table shows that GMM-LR-LM improves the correction obviously. Comparison to the traditional rule-based method, the recall rate and F-score have been improved greatly, the accuracy also has a certain promotion. Meanwhile, compared with the third-party input method (tens of millions of users), the indicators of new auto-correction model are better, but the method of the third-party input method is not published. However, the wrong rate is still above 5%，these errors affect the user experience very much. We need to consider more advanced models to improve accuracy and recall.

In the model, some of the features are assumed to be independent, ignoring their internal relations, but in practice this is not good assumption. Therefore, more advanced models can be considered and the potential relationships between these features can be constructed. For example, the GMM model and the LR are calculated separately and can be combined into a higher-dimensional GMM. In addition，the features are manually selected, we can consider using deep learning models [6] to introduce more hidden features.

4 Conclusion

In this research，we propose the GMM-LR-LM auto-correction model for flick keyboard input. The model adopts features originated from touch screen interaction and context features from language model to perform error correction. The performance of auto-correction has been greatly improved. Our future work will continue to optimize the model and to learn the local self-adaption of auto- correction.

Reference

[1] Kukich K. Techniques for automatically correcting words in text[J]. ACM Computing Surveys (CSUR), 1992, 24(4): 377-439.

[2] Wagner R A, Fischer M J. The string-to-string correction problem[J]. Journal of the ACM (JACM), 1974, 21(1): 168-173.

[3] Cai F, De Rijke M. A survey of query auto completion in information retrieval[J]. Foundations and Trends® in Information Retrieval, 2016, 10(4): 273-363.

[4] Ding Q, Han J, Zhao X, et al. Missing-data classification with the extended full-dimensional Gaussian mixture model: Applications to EMG-based motion recognition[J]. IEEE Transactions on Industrial Electronics, 2015, 62(8): 4994-5005.

[5] Menard S. Applied logistic regression analysis[M]. Sage, 2002.

[6] Zhao, Dan, and Jian Sun. "Research on the Automatic Error Correction Model Combined with Artificial Intelligence for College English Essays." Journal of Residuals Science & Technology 13.5 (2016).