A Chinese Mobile Phone Input Method Based on the Dynamic and Self-study Language Model

Zhu Qiaoming, Li Peifeng, Gu Ping, Qian Peide

School of Computer Science & Technology of Soochow University，Suzhou，215006
{qmzhu, pfli, pgu ,pdqian}@suda.edu.cn

Abstract. This paper birefly introduces a Chinese digital input method named as CKCDIM (CKC Digital Input Method) and then applies it to the Symbian OS as an example, and it also proposes a framework of input method which adopted the Client/Server architecture for the handheld computers. To improve the performance of CKCDIM, this paper puts forward a dynamic and self-study language model which based on a general language model and user language model, and proposes two indexes which are the average number of pressed-keys (ANPK) and the hit rate of first characters (HRFC) to measure the performance of the input method. Meanwhile, this paper brings forward a modified Church-Gale smoothing method to reduce the size of general language model to meet the need of mobile phone. At last, the experiments prove that the dynamic and self-study language model is a steady model and can improve the performance of CKCDIM.

Key Words: Chinese Digital Input Method, Architecture of Input Method, Dynamic and Self-study Language Model, HRFC, ANPK

1 Introduction

With the developing of communication technology and the popularization of the mobile phone in China, the use of text message in mobile phone is growing rapidly. According to CCTV financial news report, the total number of Short Message Service use will grow from 300 billions in 2005 to 450 billions in 2006 in China. However, unlike in the PC platform, mobile phones including other handheld and ubiquitous computers are not naturally suited to text input both for English and for Ch inese as the standard (ISO/IEC 9995-8[1]) layout of a mobile phone uses twelve to fifteen keys to allow basic input. So there are many researchers to study how to invent an efficient text input method for those handheld and ubiquitous computers.

Masui[2] proposed an efficient text input technique called POBox (predictive compositon based on example) which can be used in various environments where conventional keyboards are difficult to use. It described the design scheme of HCI (human computer interface) and the idea of predictive text input method by showing the POBox architecture. But it did not expound the implement method. Mackenzie[3] proposed to use probabilities of letter sequence to guess the intended letter named LetterWise method. Although it focused on English text input method, we can learn that probabilities of letter sequence are important to improve the performance of text input method. Butte[4] gave the evaluation of mobile phone text input method though three different input scheme. It found that the different input scheme would get a different entry speed. To implement a university text input method, many systems support an input method framework. Sun[5] mentioned that the Java 2 platform includes an input method framework(J2IMF) to enable text input components to receive their input from different text input method. Microsoft[6] proposed the text services framework(TSF) in the current text input solution in Microsoft Windows. Both the J2IMF and TSF need the supports from their own environment. Hiura[7] mentioned an architecture for a goal of building a system that is easy to integrate with existing text input architectures and of utilizing the services that they can offer over the network, named IIIMF, that is the internet/intranet input method framework. In order to improve the performance of an input method, language model is used in most of systems. Zheng[8] and Ling[9] proposed to use statistical language model to Chinese Pinyin input and it founded that the language model can obtain a good conversion accuracy. Also Jianfeng[10] mentioned to improve language model size reduction using better pruning criteria. From above mentions, we learn that there are many factors to be considered for implementing a practical input method in mobile phone.

The rest of paper is structured as follows: Section 2 introduces briefly a Chinese digital input method named CKCDIM. Section 3 describes how to apply CKCDIM to the most popular mobile phone OS-Symbian as an example, and also gives a general framework of input method which adopted the C/S architecture for the handheld computers. To improve the performance of CKCDIM, the paper puts forward two concepts which are the average number of pressed-keys (ANPK) and the hit rate of first characters (HRFC) to measure the performance of the input method and makes a statistic for ANPK via CKCDIM with T9 Pinyin and T9 Bihua[11]. Section 4 proposes a dynamic and self-study language model which is based on a general language model and user language model. Meanwhile, the paper brings forward a modified Church-Gale smoothing method to reduce the size of general language model to meet the request of mobile phone. Section 5 presents conclusions and our future work.

2 CKC Encoding Scheme

The CKCDIM(CKC Digital Input Method) is based on an encoding scheme which named CKC Encoding Scheme. And the CKC Encoding Scheme is designed to provide the best of both the learning curve and input speed. This encoding scheme is a digital keyboard input one. Actually, it only uses a maximum of 4 digits (“0”–“9”) to represent a Chinese character. Strokes of all Chinese characters are classified into 10 groups, each represented by a digit from “0” to “9”. Chinese characters can then be inputted by the combination of these digits which are called as input codes. The detailed information can be founded in URL http://ckcsys.com.hk/introduce.htm.

3 CKC Encoding Scheme applied to Symbian OS

Symbian, the most popular mobile phone OS, provides an interface to design input method for the developers and it is called FEP(Front-End Processor). FEP is the interface between user and application, and it receives the input messages from user, and then transmits them to application. So the input method in the Symbian is a dynamic link library which implements the FEP interface actually. Other than the Latin language, Chinese languages have a big set of characters, such as 6763 common characters in GB2312-80 in mainland of China and 13053 characters in CNS 11643-1986 in Taiwan which all have used by the Chinese mobile phone in the Simplify version or Traditional version. Almost each input method needs a code table to translate the digit serial(input code) to Chinese characters, and the total size of such code table is very large for a mobile phone because its memory is so limited.

Otherwise, Symbian is a multi-tasking OS. There are many co-existing running processes in the system at the same time. And every running process must have an instance of input method which runs in its own process space. So the total memory size of all instances is very large for a small mobile phone. To solve this problem, the code table must be shared by all instances of the same input method in the memory.

3.1 The model of CKCDIM

The model of CKCDIM is based on the input method model in the Symbian OS. It adopts the Client/Server architecture to share the code table in the memory for all instances of the input method. The client is named as FEPClient and it's a DLL which has implemented the FEP interface. The FEPClient runs in the process space of the application and communicate with the user and application. The server is called as FEPServer and it is an independent background process and its task is to search and manage the code table for the FEPClient. In the Symbian, there is only one running FEPServer at any time, but probably there co-exist many running FEPClients because every process which need input character must have a FEPClient. So the relation between FEPClients and FEPServer is many-to-one. The figure 1 shows the architecture of input method in Symbian. This architecture also can be used in other OS, such as embedded Linux, Windows CE and Plam OS.

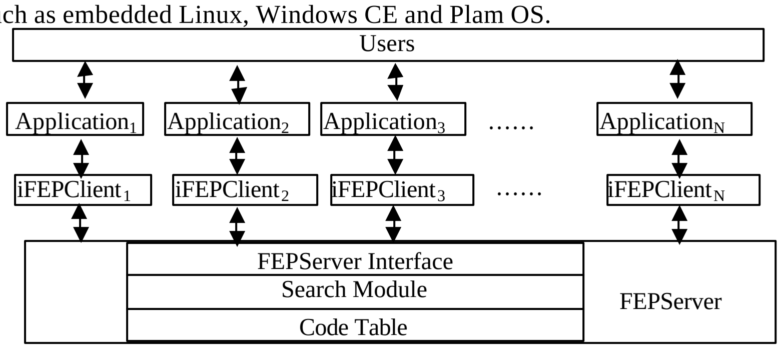

Fig. 1. The C/S architecture of input method

In the figure 1, iFEPClient$_k$($1<=k<=N$) represents an instance of input method. The main functions of iFEPClient are:

- Receive the keyboard messages and than translate it to digits;
- Interpret the mean of each digit and process it following the current status;
- Display the output candidate lists and inputted digits at the same time.

FEPServer consisted of three parts: FEPServer Interface, Search Module and Code Table. The FEPServer Interface provides a set of call-back functions for iFEPClient used to translate the input codes to Chinese characters. The Search Module realizes the algorithms to search the code table and translate input keys to Chinese characters. To support cooperative multi-tasking, the AO (Active Object) is introduced to realize FEPServer. AO is a paradigm for non-pre-emptive multi-tasking within a single-threaded application. When the FEPServer starts, it will load a part of the code table if necessary, such as index table. Then the FEPServer will start the Active Scheduler (AS) and begin to listen the request from FEPClient.

3.2 Compare with the other Input Method in the mobile phone

Two kinds of data were collected before handling for the evaluation experiment. We have collected the code table of T9 Pinyin, T9 Bihua and CKCDIM based on the character set GB2312 which including 6763 Chinese characters. Firstly, we calculate the average length of all input codes in the code table. The average length of all input codes is defined as follows:

$$\mathrm{ALIC} = \frac{\text{the sum of the length of all Chinese character's input code}}{\text{the sum of all Chinese characters in code table}} \qquad (1)$$

ALIC means the average length of all input codes, so the lower ALIC is, the faster the input method is. The table 1 shows the static ALIC of three input methods.

Table 1. The ALIC of three input methods (GB2312, 6763 Chinese characters)

Input Method	ALIC(PC keyboard)	ALIC(digital Keyboard)
CKCDIM	3.334	3.334
T9 Pinyin	3.081	6.532
T9 Bihua	10.665	10.665

In table 1, left line of ALIC shows the average length of input codes using a PC keyboard with 102 keys and the right line shows it used a digital keyboard with 18 keys. The results show that the T9 pinyin has the lowest ALIC when the users use PC keyboard to input Chinese characters, while CKCDIM is the lowest ALIC when the users use a digital keyboard. CKCDIM and T9 Bihua encoded the input code with digits (0-9), so the ALIC is same in PC and mobile phone. But T9 pinyin encoded the input code with English characters (a-z) in PC. When T9 pinyin is introduced to mobile phone, it must map the English characters to digits and the relation is many to one. Therefore, the ALIC increased to 6.532 from 3.081 after the Pinyin was encoded with digits.

In fact, a user needn't input all of the input code to input the Chinese characters and some characters maybe are used frequently and other characters maybe are seldom used, so we can not infer the input speed only using ALIC. Therefore, we must use another dynamic experiment to evaluate the input speed. This experiment provides a set of short messages as the input texts and then counts the total numbers of key-pressed(including all sorts of the keys, such as input code, select-key, arrow-key, switch-key, confirm-key, etc.) by each input method to input those texts.

There are two conceptions to show the performance of the input method: ANPK and HRFC. They are defined as follows:

$$\text{ANPK} = \frac{\textit{the sum of the pressed} - \textit{key}}{\textit{the sum of the inputted Chinese characters}} \tag{2}$$

$$\text{HRFC} = \frac{\textit{the sum of the inputted characters in the first place of cadidatelist}}{\textit{the sum of inputted characters}} \tag{3}$$

ANPK means the average pressed-key number to input a Chinese character, so the lower ANPK is, the faster the input method is. HRFC means the probability of a Chinese character/phrase appears in the first place in the input method's candidate list. The higher the probabilities of the expected character/phrase on the first place are, the more convenient the user is (the user needn't use the arrow key to choose if the expected character/phrase are on the first place).

We collected 1200 short messages from websites as the second experiment data. The average length of those short messages is 32 Chinese characters. We simulate the input action by an evaluation program and the results of ANPK in this dynamic experiment are showed as table 2. In our experiment, we used all functions which provided by the input methods (such as associate-function) to reduce ANPK, so the value of ANPK in this experiment is lower than before. As the Table 2 showed, CKCDIM has the lowest ANPK, and this means the user can input fewer keys to input the same Chinese characters when he uses CKCDIM.

Table 2. the results of ANPK in the dynamic experiment (1200 short messages)

Input Method	CKCDIM	T9 Pinyin	T9 Bihua
ANPK	3.012	4.066	3.727

4 A Dynamic and Self-study Language Model

Before we discuss the dynamic and self-study language model, we firstly look into a general language model and a user language model. Then we combined with them to form a dynamic and self-study language model.

4.1 General language model

A language model is usually formulated as a probability distribution *P(s)* over strings *s* that attempts to reflect how frequently a string *s* occurs as a sentence. In an input method, a sentence is inputted phrase by phrase, so if the input method could predict the next input phrase from the current input phrase, it could place this phrase on the first place of candidate list so that the user could input this phrase faster and easier than before. We notices that for a sentence *s* composed of phrases $w_1 \dots w_n$, and this prediction can be express as probability $P(w_n|w_1w_2 \dots w_{n-1})$ where $1=i=n$. This model uses *n-1* predecessor phrases to predict the *nth* phrase and is called n-gram. The probability *P* can be estimated by Maximum Likelihood Estimate(MLE), but MLE overestimates the occurred events and underestimates the un-occurred events, so it may cause the problem of data sparseness. To solve this problem, we should introduce smoothing technique to reassign the probabilities of all events-assigning part of probabilities of occurred events to un-occurred events.

The general language model is a static statistic model based on bi-gram and the Church-Gale smoothing is used to smooth the probabilities of all null probabilities.

Our corpus consists of 7,400,000 pieces of news, short message, e-mail and film dialogue from about 100 website and the total characters number of that corpus is 750 million. At last we can get 300 million Chinese phrases after word segmentation.
The processing of building such general language model is as follows:

(1) Document pre-preparing

Firstly, these html documents are downloaded from Internet via a crawl robot. Then they are converted to plant text by a converting tool. At last, all the phrases which would influence the correctness of the probabilities such as ads and newspaper names are deleted from the corpus by rules.

(2) Word segmentation

The second step is word segmentation. In order to improve the precision we design and implement a word segmentation tool named CKCWST(CKC Word Segmentation Tool)[12], and it is used to segment the paint text to single phrase.

(3)Calculating the probabilities of bi-gram

The last step is using CMU SLM Toolkit to calculate the probabilities of each bi-gram items. To reduce the size of bi-gram model, we only calculate the bi-gram probabilities of 60 thousand high-frequent phrases.

The size of phrases lib is 65384 items, so the number of all of the item’s probabilities that are been calculated from the corpus is about 16.97 million. If each item needs 8 bytes to store, the size of this bi-gram model is about 129MB. It’s too big for an input method used in mobile phone, so we should reduce the size and smooth the probabilities to meet such request.

(4) Modified Church-Gale smoothing

Church and Gale described a smoothing method that combined Good-Turing estimate with a method for merging the information from lower- and higher-order models[13].
We describe this method for a bi-gram model. To simplify this method, consider using the Good-Turing estimate directly to build a bi-gram distribution. After the Good-Turing estimate, we have got a good general language model for input method.

But considering the size (16.97 million) of that model, we find it doesn't meet the need of the mobile phone. So we should reduce the size of that model. We pick up some items that are not so important for our model and assume them as un-occurred events and assign their probabilities to all un-occurred events. The rule to judge the item's importance that we used is whether it belongs to independent events. If it belongs to the independent events, we should pick up them as un-occurred events.
If we assume that bi-grams w_{i-1}^{i} are independent events, we have

$$p_{MLE}(w_{i-1}^{i}) = p_{MLE}(w_i)\, p_{MLE}(w_{i-1}) \qquad (4)$$

Now we should find out the items that don't accord with above assumption. We consider that the events correspond to normal scope according to the central limit theorem, so we use the *t* Test to check above assumption. The value of *t* is

$$t = \frac{\bar{x} - m}{\sqrt{\frac{s^2}{N}}} \qquad (5)$$

where
Sample Mean $\bar{x} = p_{MLE}(w_{i-1}^{i})$
Distribution Mean $m = p_{MLE}(w_i)\, p_{MLE}(w_{i-1})$
Sample Variance $s^2 = \bar{x}\,(1 - \bar{x})$
N is the size of the samples, N=16,970,000.
For each bi-gram item's value, the bigger it is, the more important it is. Using above formula, we can calculate all item's *t* values, and sort them following their values from big to small. Then we store previous *nth* items to the bi-gram model. The value of *n* depends on the request size of the bi-gram model. Because we have given up some small value probabilities, the probabilities should be smoothed again.

The buckets is partitioned according to $p_{MLE}(w_i)p_{MLE}(w_{i-1})$ from the above formula, and the sum of the probabilities which weren't stored in the bi-gram model approach to $p_{MLE}(w_i)p_{MLE}(w_{i-1})$. So the un-occurred events weren't assigned with average probabilities, but according to the values of $p_{MLE}(w_i)\, p_{MLE}(w_{i-1})$. Then

$$p_b(w_{i-1}^{i}) = l_b\, p_{MLE}(w_i)p_{MLE}(w_{i-1}) \qquad (6)$$

where

$$\sum_{all\ un-occurred\ events\ in\ bucket\, b} l_b p_{MLE}(w_i)p_{MLE}(w_{i-1}) = p_0^b \qquad (7)$$

can be used to calculate the probabilities of un-occurred events in the bucket *b*. p_0^b is the sum of the probabilities of the un-occurred events in the bucket *b* after reduced the size of bi-gram model. So the sum of all probabilities equal to 1 and the un-occurred events are smoothed finely.

Our experiments show that if the size of that model reduced to 5% (about 1.2M), the performance of the input method (including ANPK and HRFC) is also good enough. So we consider the method to reduce the size of the model is useful and it's efficient to smooth un-occurring events.

4.2 User Language Model

The user language model uses the special-user corpus, which comes from the user historical inputted text. In the input method we have recorded the $c(w_i)$ and $c(w_{i-1}^{i})$ during the user input processing. So the user language model will be changed even if the user just inputs one Chinese phrase. At initial statement the corpus is empty. And it will gradually expand with the user's input actions. But in general the size of that special-user corpus is very small. So we adopt Katz smoothing method to solve the data sparseness problem of the small special-user corpus[14].

4.3 The dynamic and Self-study language model

How to weigh the relative reliability between general language model and user language model is the most important part in the design of such dynamic and self-study language model. User language model will become more and more important with the increase of the user's historical inputted text. Yet not every item in the user model has the same reliability. In this part, we give a method of constructing the dynamic and self-study language model based on combining of the general language model and user language model.

Toward a great deal of events, we experiment by calculating the MLE of the events when the total statistical data increasing ceaselessly. The result indicates the amount of the data have no effect on k in evidence. That is to say for a given event, the probability goes steady as long as it appears more than k, in spite of how small the total statistical data is. Figure 2 shows the MLE movement of part events when the events appear from 1 to 40 times.

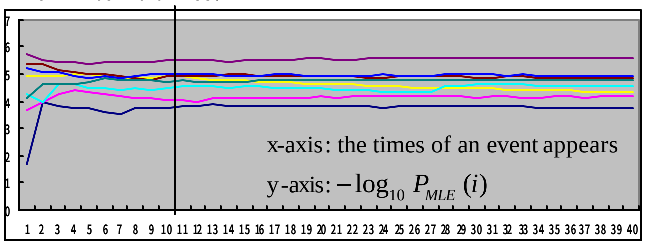

Fig 2 The probability of MLE is becoming steadier while the time of an event appears increasing.

From figure 2, it's clearly that with the increasing of the event's appearance, every line goes steady. Moreover almost every line becomes a horizontal line when the event appears more than 5(recommended by Katz) times. Therefore we can absolutely believe the event's probability through calculating the MLE when they appear more than 5 times in the user language model. And when they appear from 0 to 4 times in the user language model, Katz smoothing method is used to calculate the probability. Still it's necessary to combine the general language model and user language model because the sparseness problem of user data.

Although we have smoothed the probabilities in the general language model and got a high reliability, it still can't satisfy the own need of every user. And there is only a limit amount of data in the user language model, so the reliability is very bad toward the unobserved events and observed events that have appear few times. In this paper we shape a new model by combining the general language model with the user language model. *$P_G(i)$* is defined as the probability of event *i* in the general language model, and *$P_U(i)$* is defined as the probability of event *i* in the user language model. So *$p(i)$* can be defined as follow

$$p(i) = \frac{(1-l_r)p_G(i) + l_r p_U(i)}{\sum_{r\geq 0, i\in I}(1-l_r)p_G(i) + \sum_{r\geq 0, i\in I} l_r p_U(i)} \quad \textbf{(8)}$$

(I is the sets of all events)

to calculate the probability of event *i* in the dynamic and self-study language model. We consider toward an event *i* in the user language model, the more times it appears, the more high reliability *$P_U(i)$* has. When r>=k (k=5), set $l_r = 1$. For the unobserved events in user language model, we have a good reason to adopt absolutely distrusted policy, and then let $l_0 = 0$. For $0 < r < k$ we set $l_1 = 0.2$, $l_2 = 0.4$, $l_3 = 0.6$ and $l_4 = 0.8$ respectively.

In this way we complete the coupling of the general language model and user language model, and we call the new model as dynamic and self-study language model.

4.4 Experiment data and performance analysis

To test the performance of the language model, we should add such model to CKCDIM. The dynamic and self-study language model was inserted into the FEPServer in figure 1 and it is located in the middle of Search Module and FEPServer Interface. The dynamic and self-study language model is used to sort the candidate list by the probabilities between the current inputted phrase and the candidate phrases. So the phrase with the biggest probability will appear in the first place and the user can input it directly.

We selected 3,450 short messages from 20 websites and used them to have an input method testing. All of the short messages are divided into 5 files and each file includes 710 short messages. Firstly we split them into the Chinese phrases by our own split tool[12]. Then we calculate ANPK and HRFC by our evaluation program in mobile phone platform. In the experiments we only test Chinese characters, and non-Chinese characters, such as letters and punctuations are excluded.

Firstly we test our former CKCDIM, and then test the new CKCDIM which using the following four methods, probabilities of unigram model, general language model, move to front method, dynamic and self-study model. The unigram model, general language model and move to front method are used widely in input method on

PC platform. Figure 3 shows the change of HRFC, and Figure 4 shows the change of ANPK.

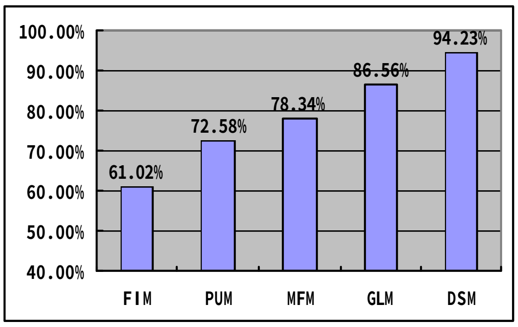

Fig 3 The HRFCs of five input methods: FIM(Former CKCDIM), PUM(CKCDIM with Probability of Unigram Model), MFM(CKCDIM with Move to Front Method), GLM(CKCDIM with General Language Model) and DSM(CKCDIM with Dynamic and Self-Study Language Model).

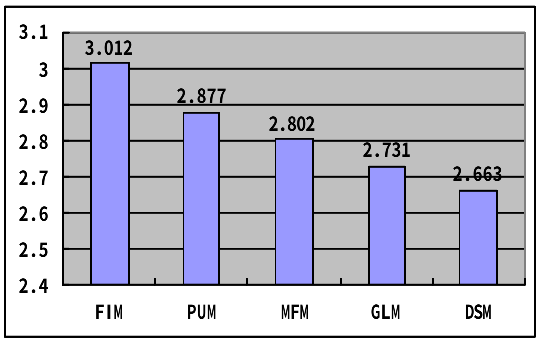

Fig 4 The ANPKs of five input methods

Applying the bi-gram model (the general model and the dynamic and self-study language model are both based on bi-gram model), The ANPK is much less than that of the original CKCDIM, meanwhile the HRFC is up to 94.23%, greatly increased than ever before. It shows the bi-gram language model get a satisfying outcome.

To test the performance of the dynamic and self-study language model, we collected 675 E-mail from five users (A、B、C、D and E) as test data. For each user, the test data include about 145 thousand Chinese characters, and we calculate the HRFC and ANPK, while every thousand Chinese characters inputted. Figure 5 shows the HRFC, and Figure 6 shows the ANPK.

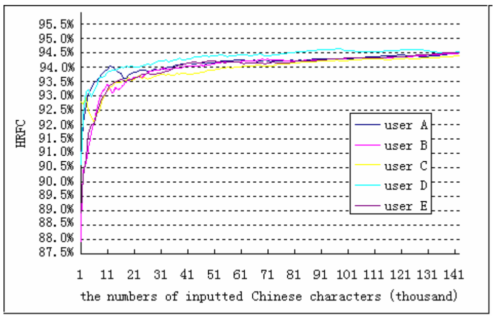

Fig 5 The relation between HRFC and numbers of inputted Chinese Characters

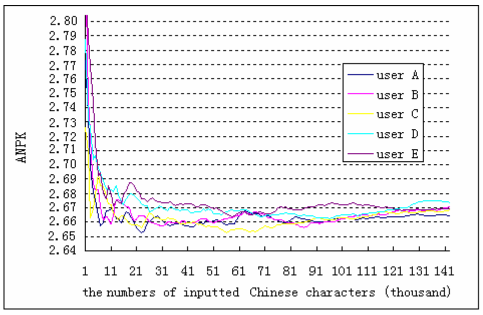

Fig 6 The relation between ANPK and numbers of inputted Chinese Characters

From figure 5, we know that the HRFC will ascend with the numbers of inputted Chinese characters increasing from 1,000 to 145,000. With the increasing of the numbers of inputted characters, every line goes steady at last. Moreover almost every line becomes a horizontal line when the numbers of inputted characters are more than 41,000.

From figure 6, we also know that the ANPK will descend with the numbers of inputted Chinese characters increasing from 1,000 to 145,000. With the increasing of the numbers of inputted characters, every line also goes steady at last.

From figure 5 and figure 6, we can find out that the performance of dynamic and self-study language model is improved rapidly by the increasing of input characters number at first and it will goes steady at last. So it is a steady and effective language model for input method.

5 Conclusions and future work

The number of mobile phone user increased rapidly in China in the recent years, while the mobile phone still can't provide a convenient and rapid-speed input method for users. This paper proposes a new input method based on a dynamic and self-study language model which can be used in mobile phone. And the experiment results show that its input speed is faster than current existing input methods. The techniques and architecture we used to design such input method also can be used in other input methods and other handhold devices.

However, the data used to build the dynamic and self-study language model is not small enough to fit all types of mobile phone, so our future work includes reducing the size of language model and optimizing such model.

Acknowledgments. The authors would like to thank the inventor of CKC encoding scheme Mr. C.K. Chou and three anonymous reviewers for their comments on this paper. This research was supported by the High Technology Plan of Jiangsu Province, China under Grant No.2005020 and the key lab of computer information processing technology of Jiangsu Province.

References

1. ISO/IEC 9995-8.: Information technology -- Keyboard layouts for text and office systems -- Part 8: Allocation of letters to the keys of a numeric keypad. International Organization for Standardization. (1994)
2. T. Masui.: POBox: An efficient text input method for handheld and ubiquitous computers. In Proceedings of the International Symposium on Handheld and Ubiquitous Computing (HUC'99). (1999) 289–300,
3. I. Scott MacKenzie, Hedy Kober, Derek Smith, Terry Jones, Eugene Skepner.: LetterWise: prefix-based disambiguation for mobile text input. In Proceedings of the 14th annual ACM symposium on User interface software and technology. (2001)
4. Lee Butts, Andy Cockburn.: An evaluation of mobile phone text input methods. Third Australasian conference on User interfaces. Melbourne (2002) 55-59
5. Sun.: Java 2 input method framework. http://java.sun.com/j2ee/1.4/docs/guide/imf/. (2003)
6. Microsoft.: Text services framework. http://msdn.microsoft.com/library/en-us/tsf/tsf/text_services_framework.asp. (2003)
7. Hiura,H.: Internet/intranet input method architecture. http://www.li18nux.org/subgroup/im/IIIMF/whitepaper/whitepaper.html.
8. Zheng Chen, Kai-Fu Lee.: A new statistical approach to Chinese pinyin input. In Proceedings of the 38th Annual Meeting of the Association for Computational Lingu istics. Hong Kong (2000) 241-247
9. L. Jin, G.-Q. Wu, F. Zheng, W.-H. Wu.: Improved strategies for intelligent sentence input method engine system. International Symposium on Chinese Language Processing (SCSLP ' 00). (2000) 247-250
10. G. Jianfeng, Z. Min.: Improving language model size reduction using better pruning criteria. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (2002) 176-182
11. Tegic Communications Inc.: T9 text input. http://www.tegic.com
12. Tao Wen, Qiao-ming Zhu, Qiang lv.: A fast algorithm for Chinese word segmentation. Computer Engineering, Vol. 30. (2004) 119-120
13. Good, I. J.: The population frequencies of species and the estimation of population parameters. Biometrika, Vol. 40. (1953) 237-264.
14. Katz, Slava M.: Estimation of probabilities from sparse data for the language model component of a speech recognizer. IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 35. (1987) 400-401