CHIME: An Efficient Error-Tolerant Chinese Pinyin Input Method

Yabin Zheng[1], Chen Li[2], and **Maosong Sun[1]**
[1]State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
[2]Department of Computer Science, University of California, Irvine, CA 92697-3435, USA
{yabin.zheng,sunmaosong}@gmail.com, chenli@ics.uci.edu

Abstract

Chinese Pinyin input methods are very important for Chinese language processing. In many cases, users may make typing errors. For example, a user wants to type in "shenme" (什么, meaning "what" in English) but may type in "shenem" instead. Existing Pinyin input methods fail in converting such a Pinyin sequence with errors to the right Chinese words. To solve this problem, we developed an efficient error-tolerant Pinyin input method called "CHIME" that can handle typing errors. By incorporating state-of-the-art techniques and language-specific features, the method achieves a better performance than state-of-the-art input methods. It can efficiently find relevant words in milliseconds for an input Pinyin sequence.

1 Introduction

Chinese uses a logographic writing system, while English and many other romance languages use an alphabetic writing system. Users cannot type in Chinese characters directly using a Latin keyboard. Pinyin input methods have been widely used to allow users to type in Chinese characters using such a keyboard. Pinyin input methods convert a Pinyin sequence to the corresponding Chinese words, where Pinyins are used to represent pronunciations of Chinese words. For instance, suppose a user wants to type in the Chinese word "上海" (the city Shanghai). He types in the corresponding Pinyin "shanghai", and the input method displays a list of Chinese words with this pronunciation, as shown in Figure 1 (left). The user selects the word "上海" as the result.

As users of other languages, Chinese users often make errors when typing in a Pinyin sequence. Existing Pinyin-based methods have error-tolerant features such as fuzzy tone for Chinese southern dialects. However, there are many input errors that cannot be handled by these methods. In our example of "shanghai" (上海), the user may make typos, thus type in "shanghaai" instead. As shown in Figure 1 (right), typical methods cannot return the right word. This limitation affects user experiences greatly, since he has to identify and correct the typos, or cannot find the right word.

In this paper, we study how to automatically detect and correct typing errors in an input Pinyin sequence, and convert it to the corresponding Chinese words. In the running example, for the mistyped Pinyin "shanghaai", we can find a similar Pinyin "shanghai", and suggest Chinese words including "上海", which is indeed what the user is looking for.

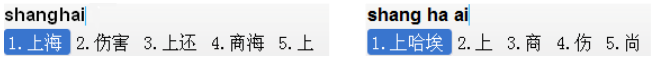

Figure 1: Typical Chinese Pinyin input method for a correct Pinyin (left) and a mistyped Pinyin (right).

When developing an error-tolerant Chinese input method, we are facing two main challenges. First, Pinyins and Chinese characters have two-way ambiguity. Many Chinese characters have multiple pronunciations. The character "和" can be pronounced as "hé", "hè", "huó", "huò", and "hú". On the other hand, different Chinese characters can have the same pronunciation. Both characters "人" and "仁" have the same pronunciation "rén". The ambiguity makes it technically challenging to do Pinyin-to-Chinese conversion. Notice that it is important for us to suggest relevant words to a user, since otherwise the user may be annoyed by seeing suggested words that are irrelevant to the input sequence. Thus we need to detect and correct mistyped Pinyins *accurately*.

Another challenge is how to do the correction *efficiently*. Existing input methods suggest Chinese words as a user types in a Pinyin sequence letter by letter. Studies have shown that we need to answer a keystroke query within 100 ms in order to achieve an interactive speed, i.e., the user does not feel any delay. Achieving such a high speed is especially important when the method is used by a Web server that can receive queries from many users, and requires a method to find relevant Chinese words for a mistyped sequence efficiently.

In this paper we develop a novel error-tolerant Chinese Pinyin input method that can meet both requirements of high accuracy and high efficiency. It is called "CHIME", which stands for "CHinese Input Method with Errors." CHIME works on a Pinyin sequence as follows. For a mistyped Pinyin in the sequence that does not exist in a Pinyin dictionary, we find similar Pinyins as candidates. We rank them using a noisy channel model and Chinese-specific features to find those that are most likely what the user intends to type in for this Pinyin. Finally, we use a statistical language model on the input sequence to find the most likely sequence of Chinese words.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 analyzes typing errors at the Pinyin level. In Section 4, we study how to convert a Pinyin sequence to a sequence of Chinese words. In Section 5, we report experimental results. Section 6 concludes the paper and discusses future work.

2 Related Work

Chen and Lee [2000] used a statistical segmentation and a trigram-based language model to convert Chinese Pinyin sequences to Chinese word sequences. They also proposed an error model for Pinyin spelling correction. However, they only correct single-character errors, while our method can correct multi-character errors.

Kwok and Deng [2002] proposed a method of extracting Chinese Pinyin names from English text and suggesting corresponding Chinese characters to these Pinyins. However, their approach can only convert Pinyin names to Chinese characters, while our method can handle general words.

There are studies on spelling correction mainly for English. Brill and Moore [2000] introduce a noisy channel model for spelling correction based on generic string-to-string edit operations. Without a language model, their error model gives a 52% reduction in spelling correction error rate. With a language model, their model gives a 74% reduction in error rate. Toutanova and Moore [2002] extend the work using pronunciation information. They use pronunciation similarities between words to achieve a better accuracy. Inspired by [Toutanova and Moore, 2002], we also take pronunciation information in Chinese to gain better performance. More details are given in Section 3.

Cucerzan and Brill [2004] use information in search query logs for spelling correction. Their approach utilizes an iterative transformation of the input query strings into other correct query strings according to statistics learned from search query logs. Sun et al. [2010] extract query-correction pairs by analyzing user behaviors encoded in the click-through data, and then a phrase-based error model is trained and integrated into a query speller system. Gao et al. [2010] make significant extensions to a noisy channel model and treat spell checking as a statistical translation problem. Experiments show that these extensions lead to remarkable improvements.

Whitelaw et al. [2009] implement a system for spelling correction in various languages. Their system does not require any manually annotated training data. They adopt the Web as a knowledge base to learn misspellings and word usage. These methods are not directly applicable to the Chinese language due to the unique ways users type in Chinese words.

In this paper, we assume that a user already segments an input Pinyin sequence by explicitly typing in a space between two adjacent Pinyins. For example, suppose a user wants to type in a sentence "我们 购买 了 上海 生产 的 牛奶" (meaning "We bought milk made in Shanghai"). The correct input Pinyin sequence is "women goumai le shanghai shengchan de niunai". However, the user types in a sequence "**woemng gounai** le **sanghaai shengchang** de **niulai**". We will use this example throughout this paper.

3 Correcting a Single Pinyin

We first study the problem of how to correct a single Pinyin. Given a dictionary D of Pinyins and an input Pinyin p that is not in the dictionary, our method CHIME finds a set of candidate Pinyins $w \in D$ that are most likely to be erroneously typed in as p. First, we do a similarity search to find candidate Pinyins that are similar to the input Pinyin. Second, we rank these candidates and find those with the highest probabilities to be what the user intends to type in.

3.1 Finding Similar Pinyins

Measuring Pinyin Similarity: As shown in [Cooper, 1983], most typing errors can be classified into four categories: insertions, deletions, substitutions, and transpositions. Therefore, in this paper we use edit distance to measure the number of errors between a candidate Pinyin and a mistyped Pinyin. Formally, the edit distance between two strings is the minimum number of single-character operations required to transform one sting to the other. We generalize the definition of edit distance by allowing transpositions.

We assume an upper threshold on the number of typing errors a user can make in a single Pinyin. In other words, for a mistyped Pinyin, we assume that the edit distance between the correct Pinyin and the input one is within the threshold. Damerau [1964] showed that 80% of typing errors are caused by a single edit operation. Using a threshold of 2, we can find the correct Pinyin with a very high probability. For example, for the Pinyin "sanghaai", by finding those whose edit distance to the input is within 2, we can find candidates "shanghai", "canghai", and "wanghuai".

Efficient Similarity Search: We adopt the state-of-the-art index structure and search algorithm proposed in [Ji *et al.*, 2009] to find those similar candidate Pinyins efficiently. The main idea is to build a trie structure to index the Pinyins in the dictionary. As a user types in a Pinyin letter by letter, the algorithm can on-the-fly do a similarity search to find those trie nodes, such that the edit distances between the corresponding prefixes and the input Pinyin are within a given threshold. From the trie nodes of these similar prefixes we can find their leaf nodes as candidate similar Pinyins.

Consider the sequence in our running example, "**woemng gounai** le **sanghaai shengchang** de **niulai**". For each Pinyin, we find similar candidate Pinyins in the dictionary, whose edit distance to the given Pinyin is no greater than 2. The candidate Pinyins are shown in Table 1, where we underline each correct Pinyins that the user intends to type in.

Input Pinyin	Similar Candidate Pinyins
woemng	women, weng, wodang
gounai	goumai, dounai, guonei
le	le
sanghaai	shanghai, canghai, wanghuai
shengchang	shengchan, zhengchang, shangchang
de	de
niulai	niunai, niupai, niuli

Table 1: Similar candidates for mistyped Pinyins.

3.2 Ranking Similar Pinyins

Each Pinyin can have multiple similar candidates, and CHIME ranks them to decide which of them are most likely what the user wants. We discuss how to do ranking using a noisy channel error model and language-specific features.

Using Noisy Channel Error Model: Given a mistyped Pinyin p, in a noisy channel error model [Brill and Moore, 2000; Kernighan *et al.*, 1990], we want to rank each similar candidate p' based on its conditional probability $Pr(p'|p)$. We find those with the highest conditional probabilities as the suggestions, since they are most likely what the user wants to type in. To estimate $Pr(p'|p)$, we apply the Bayes rule and have:

$$Pr(p'|p) = \frac{Pr(p|p')Pr(p')}{Pr(p)}. \propto Pr(p|p')Pr(p') \quad (1)$$

We assume that the user first chooses a Pinyin p' to type in according to the prior probability distribution $Pr(p')$. Then the user mistakenly types in a Pinyin p with the conditional probability $Pr(p|p')$. In Formula (1), the prior probability $Pr(p')$ shows how popular this Pinyin p' is.

Estimating Conditional Probability $Pr(p|p')$: To estimate $Pr(p|p')$, we consider an optimal transformation T from p' to p that has the minimum number of edit operations. For each edit operation e in T, let $Pr(e)$ be its probability. Assuming these edit operations are independent, we take the product of their probabilities as the probability $Pr(p|p')$.

$$Pr(p|p') = \prod_{e \in T} Pr(e). \quad (2)$$

We use '$\sim$' to denote the empty string. Then, an optimal transformation T from $shanghai$ to $sanghaai$ contains two edit operations: 'h'→'$\sim$' (deleting the first 'h' letter) and '$\sim$'→'a' (inserting the letter 'a'). Then, we have: $Pr(sanghaai|shanghai) = Pr(\text{'h'}\rightarrow\text{'}\sim\text{'})Pr(\text{'}\sim\text{'}\rightarrow\text{'a'})$.

Ristad and Yianilos [1998] proposed a method for estimating the probabilities of edit operation $Pr(e)$ from a training corpus. In this paper, we focus on proposing a framework for our error-tolerant Chinese Pinyin input method. Moreover, there is no publicly available training corpus in Chinese Pinyin. Therefore, in the absence of such a training corpus, we use heuristic rules based on Chinese-specific features to determine these probabilities. Some features are shown in Table 2.

We explain these features as follows. There are many dialects in China. Many people in southern China do not speak the standard Mandarin, and they do not distinguish retroflex and blade-alveolar, as well as front and back nasal sound. In addition, some letters are pronounced similarly. For instance, letters 'n' and 'l' have similar pronunciations. For another example, many Chinese speakers tend to be confused by whether they should use the letter 'g' in a Pinyin ending with "-ing" and a Pinyin ending with "-in" due to the similarity between a front nasal sound a back nasal sound.

In summary, for a mistyped single Pinyin p, CHIME finds similar Pinyins p' and compute their conditional probabilities $Pr(p'|p)$ using Formulas (1) and (2). CHIME takes candidates with the highest probabilities as the correct Pinyins.

Feature	Example pairs of similar Pinyin letters
Front and back nasal sound	'ang' - 'an', 'ing' - 'in', 'eng' - 'en'
Retroflex and blade-alveolar	'zh' - 'z', 'sh' - 's', 'ch' - 'c'
Letters with similar pronunciations	'z' (兹) - 'c' (词) - 's' (丝), 'n' (呢) - 'l' (勒), 'b' (播) - 'p' (泼)

Table 2: Common Chinese-specific typos.

4 Converting Pinyin Sequences to Chinese Words

Considering a Pinyin sequence $P = p_1 p_2 \cdots p_k$, where each p_i is a Pinyin, we want to find the most probable sequence of Chinese words $W = w_1 w_2 \cdots w_k$, where each w_i is a Chinese word for the Pinyin p_i. Figure 2 shows different Chinese words for each Pinyin in the running example.

One simple way is to select the most likely Chinese word for each Pinyin *separately*, and concatenate these words as the result. However, it does not consider the context of each Pinyin when suggesting a Chinese word. Next we discuss how CHIME uses a language model when suggesting Chinese words. We first discuss how CHIME converts a Pinyin sequence to Chinese words when the sequence is correct.

4.1 Pinyin-to-Chinese Conversion without Typos

Suppose the input sequence P does not have typos. We want to find the most probable sequence of Chinese words $W = w_1 w_2 \cdots w_k$. In other words, we want to maximize the conditional probability $Pr(W|P)$. We want to find:

$$\begin{aligned} \hat{W} &= \arg\max_W Pr(W|P) \\ &= \arg\max_W \frac{Pr(W)Pr(P|W)}{Pr(P)} \\ &= \arg\max_W Pr(W)Pr(P|W) \\ &= \arg\max_W Pr(W) \prod_i Pr(p_i|w_i). \end{aligned} \quad (3)$$

We made the assumption of conditional independence in Formula (3). $Pr(p_i|w_i)$ is the conditional probability that a Chinese word w_i is typed in as Pinyin p_i. (Notice that w_i is a Chinese word, not a Chinese Pinyin as in Section 3.2.) $Pr(p_i|w_i)$ is 1 if p_i is a pronunciation of word w_i, and 0 otherwise.

The term $Pr(W)$ is the prior probability of a Chinese word sequence W according to a *language model* [Chen and Lee, 2000; Gao *et al.*, 2002; Jurafsky *et al.*, 2000]. To compute the probability, we make a first-order Markov assumption, and adopt a bigram language model learned from a training corpus. Then we compute $Pr(W)$ using the following formula:

$$Pr(W) = Pr(w_1)Pr(w_2|w_1) \cdots Pr(w_n|w_{n-1}). \quad (4)$$

CHIME uses maximum likelihood estimation (MLE) to compute the conditional probability $Pr(w_n|w_{n-1})$, and uses Katz backoff [Jurafsky *et al.*, 2000] for smoothing the zero probability bigrams. Finally, CHIME uses the Viterbi algorithm to find the optimal Chinese word sequence W.

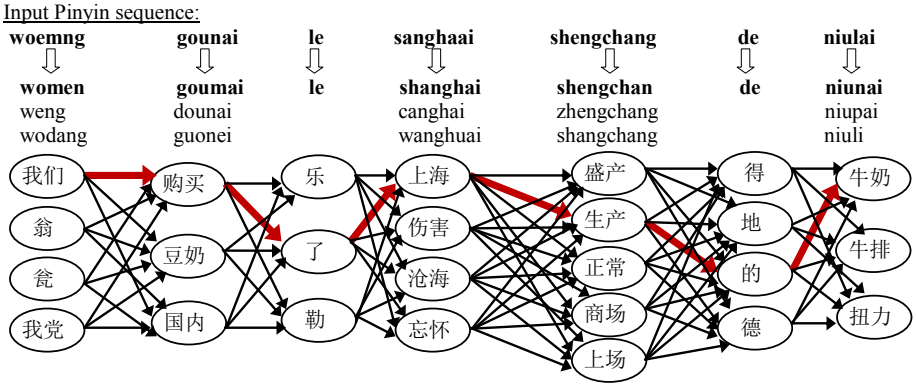

Figure 2: Correcting typos in a Pinyin sequence. The thick lines represent the final sequence of Chinese words.

4.2 Pinyin-to-Chinese Conversion with Typos

Consider the case where there are typing errors in the Pinyin sequence P. Let P' denote the correct Pinyin sequence that the user intends to type in. In our running example, $P =$ "woemng gounai le sanghaai shengchang de niulai", while $P' =$ "women goumai le shanghai shengchan de niunai". We make an assumption that given P', the input Pinyin sequence P and the optimal Chinese word sequence W are conditionally independent. We can refine Formula (3) as:

$$\begin{aligned}
\hat{W} &= \arg\max_W Pr(W|P) \\
&= \arg\max_W \sum_{P'} Pr(P'|P)Pr(W|P') \\
&= \arg\max_W \sum_{P'} \frac{Pr(W)Pr(P'|P)Pr(P'|W)}{Pr(P')} \\
&= \arg\max_W \sum_{P'} Pr(W)Pr(P|P')Pr(P'|W) \\
&= \arg\max_W \sum_{P'} Pr(W) \prod_i Pr(p_i|p'_i)Pr(p'_i|w_i), \quad (5)
\end{aligned}$$

In the new Formula (5), $Pr(p_i|p'_i)$ can be computed using Formula (2). $Pr(p'_i|w_i)$ is 1 if the Chinese word w_i has a pronunciation matching the Pinyin p'_i, and 0 otherwise. $Pr(W)$ can be estimated using Formula (4).

In summary, CHIME works as follows. First, we detect mistyped Pinyins that are not in the Pinyin dictionary. Then, for each such Pinyin, we find top-k similar candidate Pinyins using the method described in Section 3. Finally, we convert the corrected Pinyin sequence to the most likely sequence of Chinese words using Formula (5).

Considering our running example in Figure 2, for each mistyped Pinyin in the sequence, we pick top-3 most similar candidate Pinyins. We empirically selected top-3 most similar candidates because we find that most Pinyins that users intend to type in are included in these top-3 candidates. In addition, more candidates require more computational costs and bring really limited improvement on accuracy. We enumerate the Chinese words matching these candidate Pinyins. Finally, we find the most likely sequence of Chinese words: "我们 购买 了 上海 生产 的 牛奶".

5 Experiments

5.1 Settings

Our method needs a dictionary of Chinese words with their Pinyins. We also use a bigram language model as in Formula (4). In our experiments, we obtained this data from an open-source software called Sun-PinYin. It has a dictionary of 104,833 Chinese words and 66,797 Pinyins.

We used the Lancaster corpus [McEnery and Xiao, 2004] to evaluate the accuracy and efficiency of our method. Lancaster is a segmented Chinese corpus with Pinyins. After removing noisy sentences containing numbers or non-Chinese characters, we selected 2,000 sentences to do an evaluation. These sentences contained 21,861 Chinese characters and 11,968 Chinese words.

In order to evaluate CHIME on a real data set, we asked 5 native speakers to type in the 2,000 Chinese sentences and recorded their corresponding Pinyin sequences. We found that 679 sentences (34%) contained one or more typing errors. This number shows the necessity of developing an error-tolerant Chinese input method. The collected data had 885 mistyped Pinyins. We computed the edit distance between the mistyped Pinyins and their correct Pinyins that users intended to type in. 775 typing errors were caused by one edit operation, and 85 typing errors were caused by two edit operations. This result indicates that a good edit-distance threshold can be 2 when we want to find candidate Pinyins.

We assigned the probabilities of edit operations as follows. A letter can be replaced by another letter or transposed with the previous letter. We empirically assigned a probability of 0.25 to the transposition, and assigned a probability of 0.75 to all substitutions. Letters close to each other on a Latin keyboard or with similar pronunciations are more likely to be mistyped. For example, $Pr(\text{'z'}\rightarrow\text{'s'})$ tends to be larger than $Pr(\text{'z'}\rightarrow\text{'p'})$ because 'z' and 's' pronounce similarly (as shown in Table 2) and are close to each other on the keyboard. Operations on such similar letters were assigned with a relatively high probability. In the experiments, we used $Pr(\text{'z'}\rightarrow\text{'s'}) = 0.15$ and $Pr(\text{'z'}\rightarrow\text{'p'}) = 0.005$.

All the experiments were implemented using C++ compiled with a GNU compiler, and run on a computer with an AMD Core2 2.20GHz CPU and 4GB memory.

5.2 Evaluation Metrics

We want to know whether CHIME can successfully detect mistyped Pinyins, suggest the right Pinyins, and find the right Chinese words. Inspired by Whitelaw et al. [2009], we collected occurrences of the following kinds of errors:

1. E_1: A mistyped Pinyin is not detected. In our experiments, we had $E_1 = 331$.

2. E_2: A mistyped Pinyin is not suggested to the right Pinyin that users intend to type in. In our experiments, we had $E_2 = 464$.

3. E_3: A mistyped Pinyin is not converted to the right Chinese word. In our experiments, we had $E_3 = 474$.

We consider these different kinds of errors since they can have different impacts on user experiences. For instance, a mistyped Pinyin that is not detected is more frustrating than a mistyped Pinyin that is not suggested to the right Pinyin or converted to a wrong Chinese word.

E_1 pertains to the detection task, i.e., whether we can detect mistyped Pinyins. We define "Detection Error Rate" as $DER = E_1/T$, where T is the total number of mistyped Pinyins. In our experiments, $T = 885$.

We noticed that 331 mistyped Pinyins were not detected, because these mistyped Pinyins are valid Pinyins in the dictionary. For example, a Pinyin sequence "ta jianjian shiqu yishi" (他渐渐失去意识, meaning "He gradually loses consciousness") was mistyped as "ta jianjian shiqu ***yisi***". The mistyped Pinyin "yisi" is a valid Pinyin in the dictionary, which makes CHIME fail in detecting this mistyped Pinyin. A possible fix of this problem is to consider similar candidate Pinyins for every input Pinyin, not just those that cannot be found in the dictionary. However, this solution affected efficiency greatly in our experiments. We may also detect a *correct* Pinyin as wrong under this solution. The problem of detecting and correcting such typos needs future research investigation.

E_2 pertains to the correction task, i.e., whether we can correct mistyped Pinyins to the right Pinyins. We define "Correction Error Rate" as $CorrER = E_2/T$. About 133 (= $E_2 - E_1$) mistyped Pinyins were detected but not corrected to the right Pinyins, because we only selected three most similar candidates to reduce the processing time.

E_3 pertains to the conversion task, i.e., whether we can convert mistyped Pinyins to the right Chinese words. We define "Conversion Error Rate" as $ConvER = E_3/T$. About 10 (= $E_3 - E_2$) mistyped Pinyins were detected and corrected to the right Pinyins but not converted to the right Chinese words. The conversion errors were caused by the language model.

For comparison purposes, we used a commercial Chinese Pinyin input method called Sogou-Pinyin as our baseline. Sogou is one of the state-of-the-art Chinese input methods that covers about 70% of the Chinese-input-method market. We evaluated Sogou using the same data set and collected occurrences of the three types of errors described above. For Sogou, we had $E_1 = 625$, $E_2 = 807$, and $E_3 = 812$.

As we can see in Table 3, with an error rate of 37.4%, CHIME achieved a better performance on the task of detecting mistyped Pinyin than Sogou (70.62%). CHIME corrected mistyped Pinyins with an error rate of 52.43%, while Sogou achieved a correction error rate of 91.19%. Our error rate was 53.56% when converting Pinyins to Chinese words, which was much better than Sogou with a conversion error rate of 91.75%. Specifically, CHIME was able to convert 411 mistyped Pinyins to the right Chinese words, while Sogou converted only 73 mistyped Pinyins to the right Chinese words.

Metric	*DER*	*CorrER*	*ConvER*
CHIME	**37.40%**	**52.43%**	**53.56%**
Sogou	70.62%	91.19%	91.75%

Table 3: Results on the Lancaster corpus with typing errors.

5.3 Efficiency Evaluation

We also evaluated the efficiency of CHIME, and measured the additional computational costs if we enabled error-tolerant features. We assumed a user continually types in a Pinyin sequence letter by letter, and CHIME converted the Pinyin sequence to Chinese words interactively. For example, suppose a user wants to type in the Pinyin sequence "woemng gounai le sanghaai shengchang de niulai". She types in the first Pinyin "woemng" letter by letter, and CHIME will receive strings "w", "wo", "woe", "woem", "woemn" and "woemng". The other Pinyins are input similarly.

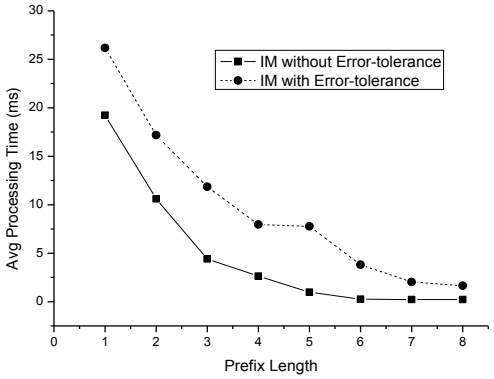

Figure 3: Processing time with different prefix lengths.

We simulated the typing process described above using 2,000 Pinyin sentences typed in by real users. The average processing time of each sentence as a complete query was 12.8ms. When users type in more letters, the number of candidate Pinyins will reduce. For example, in our compiled Pinyin dictionary, the number of candidate Pinyins which start with prefix "w", "wo", and "wom" was 2320, 92, and 3 respectively. Therefore, the average processing time decreased quickly when users typed in more letters. In Figure 3, the average processing time was 26.19 ms when the user typed in the first letter. It dropped to 17.19 ms when the user typed in the second letter.

We also recorded the performance of CHIME without the error-tolerant features, and the results are shown in Figure 3. The average processing time of each sentence was 7.83ms, which indicates that we need an additional processing time of 4.97ms if we add error-tolerant features. This shows that CHIME can achieve a very high performance.

6 Conclusion and Future Work

In this paper, we developed an efficient error-tolerant Chinese Pinyin input method called "CHIME". For a mistyped Pinyin, CHIME can find similar Pinyins. It then uses language-specific features to find most likely Pinyins. For a Pinyin sequence, CHIME detects and corrects the mistyped Pinyins, and finds the most likely sequence of Chinese words. Experiments showed that we can achieve both a high accuracy and a high efficiency compared to state-of-the-art input methods.

In the future we plan to study how to solve the problem when a user types in an entire Pinyin sequence instead of typing each Pinyin and selecting a Chinese word. Another future direction is how to accurately detect and correct a mistyped Pinyin that happens to be in the Pinyin dictionary. Secondly, a major limitation of our approach is that it is dependent on a large Pinyin dictionary. How to type in out-of-vocabulary words like named entities is also a future direction. Thirdly, how to support acronym Pinyin input (using the first letters of a syllable in Pinyin, e.g., "zg" for "中国") is also an interesting and challenging problem and needs more research.

Acknowledgments

Most of the work was done while Yabin Zheng was visiting UCI. We thank Alexander Behm and Shengyue Ji for their insightful discussions at UCI. This work is partially supported by the National Natural Science Foundation of China (No. 60873174 and 60828004).

References

[Brill and Moore, 2000] E. Brill and R.C. Moore. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 286–293. Association for Computational Linguistics, 2000.

[Chen and Lee, 2000] Z. Chen and K.F. Lee. A new statistical approach to Chinese Pinyin input. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 241–247. Association for Computational Linguistics, 2000.

[Cooper, 1983] W.E. Cooper. *Cognitive aspects of skilled typewriting*. Springer-Verlag, 1983.

[Cucerzan and Brill, 2004] Silviu Cucerzan and Eric Brill. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 293–300. Association for Computational Linguistics, 2004.

[Damerau, 1964] F.J. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964.

[Gao *et al.*, 2002] J. Gao, J. Goodman, M. Li, and K.F. Lee. Toward a unified approach to statistical language modeling for Chinese. *ACM Transactions on Asian Language Information Processing (TALIP)*, 1(1):3–33, 2002.

[Gao *et al.*, 2010] Jianfeng Gao, Xiaolong Li, Daniel Micol, Chris Quirk, and Xu Sun. A large scale ranker-based system for search query spelling correction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 358–366, 2010.

[Ji *et al.*, 2009] S. Ji, G. Li, C. Li, and J. Feng. Efficient interactive fuzzy keyword search. In *Proceedings of the 18th international conference on World wide web*, pages 371–380. ACM, 2009.

[Jurafsky *et al.*, 2000] D. Jurafsky, J.H. Martin, and A. Kehler. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. MIT Press, 2000.

[Kernighan *et al.*, 1990] M.D. Kernighan, K.W. Church, and W.A. Gale. A spelling correction program based on a noisy channel model. In *Proceedings of the 13th conference on Computational linguistics*, pages 205–210. Association for Computational Linguistics, 1990.

[Kwok and Deng, 2002] Kui-Lam Kwok and Peter Deng. Corpus-based pinyin name resolution. In *Proceedings of the First SIGHAN Workshop on Chinese Language Processing (COLING)*, pages 41–47, 2002.

[McEnery and Xiao, 2004] AM McEnery and Z. Xiao. The Lancaster Corpus of Mandarin Chinese: A corpus for monolingual and contrastive language study. *Religion*, 17:3–4, 2004.

[Ristad *et al.*, 1998] E.S. Ristad, P.N. Yianilos, M.T. Inc, and NJ Princeton. Learning string-edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(5):522–532, 1998.

[Sun *et al.*, 2010] Xu Sun, Jianfeng Gao, Daniel Micol, and Chris Quirk. Learning phrase-based spelling error models from clickthrough data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 266–274. Association for Computational Linguistics, 2010.

[Toutanova and Moore, 2002] K. Toutanova and R.C. Moore. Pronunciation modeling for improved spelling correction. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 144–151. Association for Computational Linguistics, 2002.

[Whitelaw *et al.*, 2009] C. Whitelaw, B. Hutchinson, G.Y. Chung, and G. Ellis. Using the web for language independent spellchecking and autocorrection. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 890–899. Association for Computational Linguistics, 2009.