A Unified Approach to Transliteration-based Text Input with Online Spelling Correction

Hisami Suzuki **Jianfeng Gao**
Microsoft Research
One Microsoft Way, Redmond WA 98052 USA
{hisamis,jfgao}@microsoft.com

Abstract

This paper presents an integrated, end-to-end approach to online spelling correction for text input. Online spelling correction refers to the spelling correction as you type, as opposed to post-editing. The online scenario is particularly important for languages that routinely use transliteration-based text input methods, such as Chinese and Japanese, because the desired target characters cannot be input at all unless they are in the list of candidates provided by an input method, and spelling errors prevent them from appearing in the list. For example, a user might type *suesheng* by mistake to mean *xuesheng* 学生 'student' in Chinese; existing input methods fail to convert this misspelled input to the desired target Chinese characters. In this paper, we propose a unified approach to the problem of spelling correction and transliteration-based character conversion using an approach inspired by the phrase-based statistical machine translation framework. At the phrase (substring) level, *k* most probable pinyin (Romanized Chinese) corrections are generated using a monotone decoder; at the sentence level, input pinyin strings are directly transliterated into target Chinese characters by a decoder using a log-linear model that refer to the features of both levels. A new method of automatically deriving parallel training data from user keystroke logs is also presented. Experiments on Chinese pinyin conversion show that our integrated method reduces the character error rate by 20% (from 8.9% to 7.12%) over the previous state-of-the art based on a noisy channel model.

1 Introduction

This paper addresses the problem of online spelling correction, which tries to correct users' misspellings as they type, rather than post-editing them after they have already been input. This online scenario is particularly important for languages that routinely use transliteration-based text input methods, including Chinese and Japanese: in these languages, characters (called *hanzi* in Chinese and *kanji*/*kana* in Japanese) are typically input by typing how they are pronounced in Roman alphabet (called *pinyin* in Chinese, *romaji* in Japanese), and selecting a conversion candidate among those that are offered by an input method system, often referred to as IMEs or input method editors. One big challenge posed by spelling mistakes is that they prevent the desired candidates from appearing as conversion candidates, as in Figure 1: *suesheng* is likely to be a spelling error of *xuesheng* 学生 'student', but it is not included as one of the candidates.

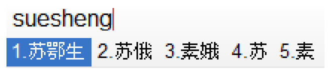

suesheng
1.苏鄂生 2.苏俄 3.素娥 4.苏 5.素

Figure 1: Spelling mistake prevents the desired output (学生) from appearing in the list of candidates

This severely limits the utility of an IME, as spelling errors are extremely common. Speakers of a non-standard dialect and non-native speakers have a particularly hard time, because they may not know the standard pronunciation of the word to begin with, preventing them from inputting the word altogether. Error-tolerant word completion and next word prediction are also highly desirable features for text input on software (onscreen) keyboards for any language, making the current work relevant beyond Chinese and Japanese.

In this paper, we propose a novel, unified system of text input with spelling correction, using

Chinese pinyin-to-hanzi conversion as an example. We first formulate the task of pinyin spelling correction as a substring-based monotone translation problem, inspired by phrase-based statistical machine translation (SMT) systems (Koehn et al., 2003; Och and Ney, 2004): we consider the pinyin input (potentially with errors) as the source language and the error-corrected pinyin as the target, and build a log-linear model for spelling correction. In doing so, we also propose a novel, unsupervised method of collecting parallel training data from user input logs. We then build an integrated end-to-end text input system that directly converts a potentially erroneous input pinyin sequence into a desired hanzi sequence, also formulated as a monotone phrase-based SMT problem, in which the feature functions of the substring-based error correction component are integrated and jointly optimized with the sentence-level feature functions for character conversion

Our method generalizes and improves over the previous state-of-the-art methods for the task of error correction and text input in several crucial respects. First, our error correction model is designed and implemented as a substring-based, fully trainable system based on a log-linear model, which has been shown effective for related tasks such as transliteration and letter-to-phone conversion, but has not been attempted for the task of spelling correction. Second, we build an end-to-end pinyin-to-hanzi conversion system by combining all the feature functions used in the error correction and character conversion components in an SMT-style log-linear model, where the feature weights are trained discriminatively for the end-to-end task. This integration method generalizes the previous approach based on a noisy channel model (Chen and Lee, 2000; Zheng et al. 2011b), in which only the error model and the conversion model probabilities are used and combined with equal weights. Finally, like other statistical systems, the amount and quality of training data control the quality of the outcome; we thus propose a new, language-independent method of deriving parallel data for spelling correction from user keystroke logs.

We performed experiments on various methods of integrating the error correction and character conversion sub-components. Our best system, a fully integrated SMT-based approach, reduces the character error rate by 35% on test data that is completely independent of the creation of error correction and character conversion models.

In what follows, we first give the background of this research in Section 2. We then describe our approach to the spelling correction task (Section 3) and the end-to-end conversion task (Section 4). We summarize our contribution and conclude with remarks for future directions in Section 5.

2 Related Work

The current work builds on many previous works on the task of monotone substring-based transduction, including spelling correction, letter-to-phone conversion and transliteration between different scripts. In particular, our substring-based approach to spelling correction is motivated by the success on transliteration (e.g., Sherif and Kondrak, 2007; Cherry and Suzuki, 2009) and letter-to-phoneme conversion (e.g., Jiampojamarn et al., 2007; Rama et al., 2009). One big challenge of the spelling correction research is the general lack of naturally occurring paired data of contextual spelling errors and their correction. Previous work has therefore either focused on the task of correcting out-of-vocabulary words out of context (e.g., Brill and Moore, 2000; Toutanova and Moore, 2002), or has resorted to innovative methods of data collection. For example, Banko and Brill (2001) generate data artificially by substituting words from a confusion word set in text for building a contextual speller; Whitelaw et al. (2009) use word frequency and edit distance information to harvest error pairs from a web corpus in an unsupervised manner; Bertoldi et al. (2010) intentionally corrupt clean text by adding noise to the data. Another approach to spelling error data collection uses web search query logs, available in large quantity (albeit to limited institutions), and limit its focus on the task of correcting misspelled queries (e.g., Cucerzan and Brill, 2004; Gao et al., 2010; Sun et al., 2010; Duan and Hsu, 2011). The problem of data collection is particularly difficult for pinyin error correction, as pinyin is not a final form of text in Chinese, so it is not recorded in final text. Zheng et al. (2011a) study a log of pinyin input method and use the backspace key to learn the user mistyping behavior, but they do so only for the purpose of data analysis, and do not build a statistical model from this data.

Text input methods have been commercially available for decades for inputting Chinese and Japanese, but have also recently become available for other non-Roman script languages including Arabic and the languages of India.[1] Early research work on text input methods includes e.g., Mori et al. (1998), Chen and Lee (2000) and Gao et al. (2002), all of which approach the problem using a noisy channel model. Discriminative approaches have also been proposed, e.g., Suzuki and Gao (2005); Tokunaga et al. (2011). There is only a very limited amount of work that deals with spelling correction in the context of text input: Zheng et al. (2011b) represents a recent work based on a noisy channel model, which defines our baseline. Their work is strictly word-based and only handles the correction of out-of-vocabulary pinyin words into in-vocabulary pinyin words, while our substring-based model is not limited by these constraints.

The current work also has an affinity to the task of speech translation in that the parallel data between the input (speech signal) and the output (text in foreign language) is not directly available, but is mediated by a corrected (transcribed) form of input. Zhang et al. (2011) is thus relevant to our study, though their approach differs from ours in that we build an integrated system that include the feature functions of both error correction and character conversion sub-systems.

3 Substring-based Spelling Correction using a Log-linear Model

In this section, we describe our approach to pinyin error correction within a log-linear framework. Though our current target is pinyin error correction, the method described in this section is applicable to any language of interest.

The spelling correction problem has been standardly formulated within the framework of noisy channel model (e.g., Kernighan et al., 1990). Let *A* be the input phonetic string in pinyin. The task of spelling correction is to search for the best correction candidate in pinyin *C** among all possible corrections for each potentially misspelled pinyin *A*:

$$C^* = \underset{C \in \mathrm{GEN}(A)}{\operatorname{argmax}} P(C|A) \qquad \dots(1)$$

Applying Bayes' Rule and dropping the constant denominator, we have

$$C^* = \underset{C \in \mathrm{GEN}(A)}{\operatorname{argmax}} P(A|C)P(C) \qquad \dots(2)$$

where the error model $P(A|C)$ models the translation probability from C to A, and the language model $P(C)$ models how likely the output C is a correctly spelled pinyin sequence. Many variations on the error model have been proposed, including substring-based (Brill and Moore, 2000) and pronunciation-based (Toutanova and Moore, 2002) models.

Our model is inspired by the SMT framework, in which the error correction probability $P(C|A)$ of Equation (1) is directly modeled using a log-linear model of the following form:

$$P(C|A) = \frac{1}{Z(A)} exp \sum_i \lambda_i h_i(A, C) \qquad \dots(3)$$

where $Z(A)$ is the normalization factor, h_i is a feature function and λ_i is the feature weight. Similarly to phrase-based SMT, many feature functions are derived from the translation and language models, where the translation model-derived features are trained using a parallel corpus of original pinyin and correction pairs. The argmax of Equation (1) defines the search operation: we use a left-to-right beam search decoder to seek for each input pinyin the best correction according to Equation (3).

We first describe how the paired data for deriving the error model probabilities is generated from user logs in Section 3.1, and then how the models are trained and the model weights are learned in Section 3.2. We discuss the results of pinyin error correction as an independent task in Section 3.3.

3.1 Generating error correction pairs from keystroke logs

Unlike English text, which includes instances of misspelled words explicitly, pinyin spelling errors are not found in a corpus, because pinyin is used as a means of inputting text, and is not part of the

[1] A few examples include Google Transliterate (http://www.google.com/transliterate/) and Microsoft Maren (http://www.microsoft.com/middleeast/egypt/cmic/maren/) / ILIT (http://specials.msn.co.in/ilit/Hindi.aspx). Quillpad (http://quillpad.in/) is also popularly used in India.

final written form of the language. Therefore, pinyin error correction pairs must be created intentionally. We chose the method of implementing a version of an input method which records the keystrokes of users while they are asked to type a particular Chinese text in hanzi; in doing so, we captured each keystroke issued by the user behind the scene. Such keystroke logs include the use of the backspace key, from which we compute the pinyin strings after the usage of the backspace keys as well as the putative pinyin string had the user not corrected it using the backspace key.[2] Table 1 shows a few examples of the entries in the keystroke log, along with the computed pinyin strings before and after correction. Each entry (or phrase) in the log represents the unit that corresponds to the sequence the user input at once, at the end of which the user committed to a conversion candidate, which typically consists of one or more words. While the post-correction string can be straightforwardly derived by deleting the same number of characters preceding the backspaces, the computation of the pre-correction string is trickier and ambiguous, because the backspace key is used for the purpose of both deletion and substitution (delete and replace) operations. In Table 1, a backspace usage is indicated by _ in the original keystroke sequence that is logged. In the second example, a deletion interpretation will generate `zhonguo` as a pre-correction string, while substitution interpretation will generate `zhonguoo`. In order to recover the desired pre-correcting string, we compared the prefix of the backspace usage (`zhonguo`) with the substrings after error correction (`zhong`, `zhongg`, `zhonggu`…). We considered that the prefix was spell-corrected into the substring which is the longest and with the smallest edit distance: in this case, `zhonguo` is considered an error for `zhongguo`, therefore recovering the pre-correction string of the whole sequence as `zhonguo`. Note that this method of error data extraction is general

keystroke	pre-correction	post-correction
n a n s _ r e n	nansen	nanren
z h o n g u o _ _ g u o	zhonguo (*zhonguoo)	zhongguo

Table 1: Computation of pre- and post-correction strings from keystroke log

and is language-independent. Since paired error correction data do not exist naturally and is expensive to collect for any language, we believe that the proposed method is useful beyond the case of Chinese text input and applicable to the data collection of the spelling correction task in general. In a related work (Baba and Suzuki, 2012), we collected such keystroke data using Amazon's Mechanical Turk for English and Japanese, and released the error-correction pairs for research purposes.[3]

The extracted pairs are still quite noisy, because one error correction behavior might not completely eliminate the errors in typing a word. For example, in trying to type *women* 我们 'we', a user might first type *wmen*, hit the backspaces key four times, retype *womeen*, and commit to a conversion candidate by mistake. We extract the pair (*wmen*, *womeen*) from this log incorrectly, which is one of the causes of the noise in the data. Despite these remaining errors, we use the data without further cleaning, as we expect our approach to be robust against a certain amount of noise.

Keystroke data was collected for three text domains (chat, blog and online forum) from 60 users, resulting in 86,783 pairs after removing duplicates. The data includes the pairs with the same source and target, with about 41% representing the case of correction. We used 5,000 pairs for testing, 1,000 pairs for tuning the log-linear model weights (see the next subsection), and the remaining portion for training the error correction component.

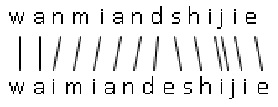
3.2 Training the log-linear model

The translation model captures substring-based spelling error patterns and their transformation probabilities. The model is learned from large amounts of pinyin-correction pairs mined from user keystroke logs discussed above. Take the

[2] Zheng et al. (2011a) also uses the backspace key in the IME log to generate error-correction pairs, but they focus on the usage of a backspace after the desired hanzi characters have been input, i.e., the backspace key is used to delete one or more hanzi characters. In contrast, our method focuses on the use of backspace to delete one or more pinyin characters before conversion. This simulates the scenario of online error correction more truthfully, and can collect paired data in large quantity faster.

[3] Available at http://research.microsoft.com/en-us/downloads/4eb8d4a0-9c4e-4891-8846-7437d9dbd869/default.aspx.

following pinyin-correction pair as an example, where the input pinyin and its correction are aligned at the character level: given a pair (A,C), we align the letters in A with those in C so as to minimize the edit distance between A and C based on single character insertions, deletions and substitutions.



From this pair, we learn a set of error patterns that are consistent with the character alignment,[4] each of which is a pair of substrings indicating how the spelling is transformed from one to another. Some examples of extracted phrases are (`wanmian`, `waimian`) and (`andshi`, `andeshi`). In our implementation, we extract all patterns with a substring length of up to 9 characters. We then learn the translation probabilities for each pair using maximum likelihood estimation (MLE). Let (a,c) denote a pair. For each pair, we learn the translation probabilities $P(c|a)$ and $P(a|c)$, estimated using MLE, as well as lexical weights in two directions following Koehn et al. (2003). Our error correction model is completely substring-based and does not use a word-based lexicon, which gives us the flexibility of generating unseen correction targets as well as supporting pinyin input consisting of multiple words at a time. For the language model, we use a character 9-gram model to capture the knowledge of correctly spelled pinyin words and phrases. We trained the language model using the target portion of the parallel data described in Section 3.1, though it is possible to train it with an arbitrary text in pinyin when such data is available.

In addition to the feature functions derived from the error and language models, we also use word and phrase penalties as feature functions, which are commonly used in SMT. These features also make sense in the current context, as using fewer phrase means encouraging longer ones with more context, and the target character length can capture tendencies to delete or insert words in errors. Overall, the log-linear model uses 7 feature functions: 4 derived from the translation models, word and phrase penalties, and the language model. The model weights were trained using the minimum error rate training algorithm (MERT, Och, 2003). We tried MERT with two objective functions: one that uses the 4-gram BLEU score as straightforwardly adapted from SMT, and the other that minimizes the character error rate (CER). CER is based on the edit distance between the reference and system output, which is used for evaluating the IME accuracy (Section 4.3). It is more directly related with the word/phrase-level accuracy, which we used to evaluate the error correction module in isolation, than the BLEU metric. As we will show below, however, using different objective functions turned out to have only a minimal impact on the spelling correction accuracy.

3.3 Experiments and results

The performance of pinyin error correction was evaluated on two data sets: (1) *log-test*: the test set of the data in Section 3.1, which is derived in the same way as the training data but is noisy, consisting of 5,000 phrases of which 2,020 are misspelled; (2) *CHIME*: the gold standard from the CHIME data set made available by Zheng et al. (2011b),[5] which is also used in the end-to-end evaluation in Section 4. This data set consists of 2,000 sentence pairs of pinyin input with errors and the target hanzi characters, constructed by collecting actual user typing logs of the Lancaster corpus (McEnery and Xiao, 2004), which includes text from newspaper, fiction, and essays.[6] The CHIME data set does not include the corrected pinyin string; we therefore generated this by running a text-to-pinyin utility,[7] and created the pairs before and after error correction for evaluating our pinyin spelling correction module. The set contains 11,968 words of which 908 are misspelled.

The results of the evaluation are given in Table 2. They are for phrase/word-level accuracy, as the log-derived data set is for each phrase (a user-

[4] Consistency here implies two things. First, there must be at least one aligned character pair in the aligned phrase. Second, there must not be any alignments from characters inside the aligned phrase to characters outside the phrase. That is, we do not extract a phrase pair if there is an alignment from within the phrase pair to outside the phrase pair.

[5] Available from http://chime.ics.uci.edu/

[6] Details on the Lancaster corpus are found at http://www.lancs.ac.uk/fass/projects/corpus/LCMC/.

[7] We used an in-house tool, but many tools are available online. Unlike pinyin-to-hanzi, hanzi-to-pinyin is relatively straightforward as most characters have a unique pronunciation.

	1-best	3-best	20-best
log-test: No correction	59.6		
log-test: Noisy Channel	49.5	67.86	84.8
log-test: Proposed (BLEU)	62.46	74.58	86.66
log-test: Proposed (CER)	62.82	75.06	86.8
CHIME: No correction	92.41		
CHIME: Noisy Channel	91.29	95.75	98.82
CHIME: Proposed (BLEU)	93.51	97.38	99.06
CHIME: Proposed (CER)	93.49	97.29	99.08

Table 2: Pinyin error correction accuracy (in %)

defined unit of conversion, consisting of one to a few words), while the CHIME data set is word-segmented. The baseline accuracy is the accuracy of not correcting any error, which is very strong in this task: 59.6% and 92.41% for the two data sets, respectively. The accuracy on the log-test data is generally much lower than the CHIME data, presumably because the latter is cleaner, contains less errors to begin with, and the unit of evaluation is smaller (word) than the log-test (phrase). Though CHIME is an out-of-domain data set, the proposed model works very well on this set, achieving more than 93% accuracy with the best output, significantly (at p<0.001 using McNemar's test) improving on the strong baseline of not correcting any error. The proposed log-linear approach is also compared against the noisy channel model baseline, which is simulated by only using one error model-derived feature function $P(A|C)$ and the language model, weighted equally, using the same beam search decoder. Somewhat surprisingly, the noisy channel model results fall below the baseline in both data sets, while the log-linear model improves over the baseline, especially on the 1-best accuracy: all differences between the noisy channel model and the log-linear model outputs are significant. Finally, regarding the effect of using the CER as the objective function of MERT, we only observe minimal impact: none of the differences in accuracy between the BLEU and CER objectives is statistically significant on either data set. For a monotone decoding task such as spelling correction, using either objective function therefore seems to suffice, even though BLEU is more indirect and redundant in capturing the phrase-level accuracy.

4 A Unified Model of Character Conversion with Spelling Correction

In this section we describe our unified model of spelling correction and transliteration-based character conversion. Analogous to the spelling correction task, the character conversion problem can also be considered as a substring-based translation problem. The novelty of our approach lies in the fact that we take advantage of the parallelism between these tasks, and build an integrated model that performs spelling correction and character conversion at the same time, within the log-linear framework. This allows us to optimize the feature weights directly for the end goal, from which from we can expect a better overall conversion accuracy.

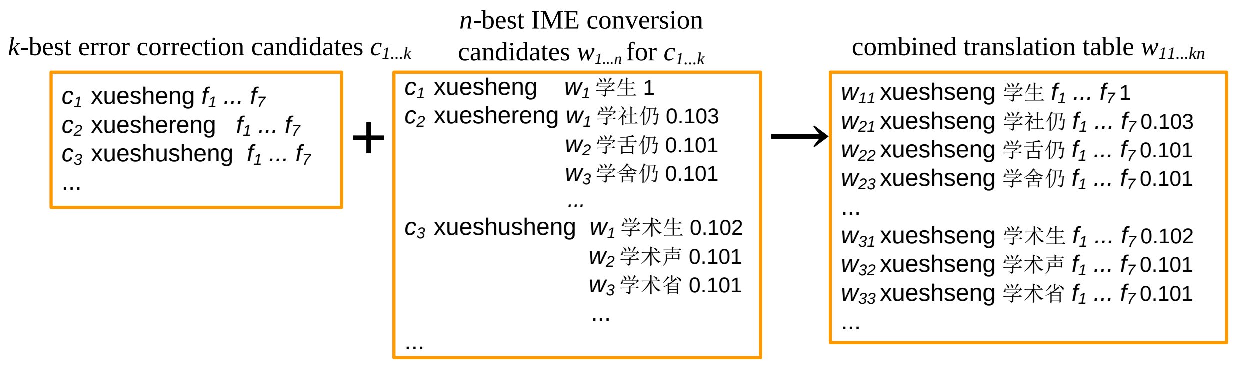
4.1 Noisy channel model approach to incorporating error correction in character conversion

The task of pinyin-to-hanzi conversion consists of converting the input phonetic strings provided by the user into the appropriate word string using ideographic characters. This has been formulated within the noisy channel model (Chen and Lee, 2000), in exactly the same manner as the spelling correction, as describe in Equations (1) and (2) in Section 3. Given the pinyin input A, the task is to find the best output hanzi sequence W*:

$$W^* \\ = \underset{W \in \mathrm{GEN}(A)}{\operatorname{argmax}} P(W|A) \qquad \ldots(4) \\ = \underset{W \in \mathrm{GEN}(A)}{\operatorname{argmax}} \mathrm{P}(W)P(A|W)$$

In traditional conversion systems which do not consider spelling errors, $P(A|W)$ is usually set to 1 if the word is found in a dictionary of word-pronunciation pairs, which also defines **GEN**(A). Therefore, the ranking of the candidates relies exclusively on the language model probability $P(W)$.

An extension of this formulation to handle spelling errors can be achieved by incorporating an actual error model $P(A|W)$. Assuming a conditional independence of A and W given the error-corrected pinyin sequence C, Equation (4) can be re-written as:



k-best error correction candidates c1…k
c1 xuesheng f1 … f7
c2 xueshereng f1 … f7
c3 xueshusheng f1 … f7
…
n-best IME conversion candidates w1…n for c1…k
c1 xuesheng w1 学生 1
c2 xueshereng w1 学社仍 0.103
w2 学舌仍 0.101
w3 学舍仍 0.101
…
c3 xueshusheng w1 学术生 0.102
w2 学术声 0.101
w3 学术省 0.101
…
…
combined translation table w11…kn
w11 xueshseng 学生 f1 … f7 1
w21 xueshseng 学社仍 f1 … f7 0.103
w22 xueshseng 学舌仍 f1 … f7 0.101
w23 xueshseng 学舍仍 f1 … f7 0.101
…
w31 xueshseng 学术生 f1 … f7 0.102
w32 xueshseng 学术声 f1 … f7 0.101
w33 xueshseng 学术省 f1 … f7 0.101
…

Figure 2: Generation of integrated translation table for the pinyin input *a* = *xueshseng*

$$W^* = \underset{W}{\operatorname{argmax}} P(W|A)$$
$$= \underset{W}{\operatorname{argmax}} \sum_C P(W|C)P(C|A)$$
$$= \underset{W}{\operatorname{argmax}} \sum_C P(C|W)P(W)P(C|A)$$

Here, $P(C|W)$ corresponds to the channel model of traditional input methods, $P(W)$ the language model, and $P(C|A)$ the pinyin error correction model. There have been attempts to use this formulation in text input: for example, Chen and Lee (2000) trained a syllable-based model for $P(C|A)$ with user keystroke data,[8] and Zheng et al. (2011b) used a model based on a weighted character edit distance whose weights are manually assigned. This noisy channel integration of error correction and character conversion is the state-of-the-art in the task of error-correcting text input, and will serve as our baseline.

4.2 Log-linear model for error-correcting character conversion

Similar to the formulation of our error correction model in Section 3, we adopt the log-linear model for modeling the character conversion probability in (4):

$$P(W|A) = \frac{1}{Z(A)} exp \sum_i \lambda_i h_i(A, W)$$

where $A = a_1,\ldots,a_n$ is a sequence of phrases in pinyin, and $W = w_1,\ldots,w_n$ is the corresponding sequence in hanzi. A unique challenge of the current task is that the parallel data for A and W do not exist directly. Therefore, we generated the translation phrase table offline by merging the substring-based phrase table generated for the pinyin error correction task in Section 3 with the results of character conversion. This process is described in detail in Figure 2: k-best candidates for each input pinyin phrase a are generated by the error model in Section 3, which are then submitted offline to an IME system to obtain n-best conversion candidates with probabilities. For the IME system, we used an in-house conversion system, which only uses a word trigram language model for ranking. In the resulting translation table, defined for each (a, w) pair, the feature functions and their values are inherited from the pinyin error correction translation table mediated by the correction candidates $c_{1\ldots k}$ for a, plus the function that defines the IME conversion probability for (c_j, w). Note that in this final phrase table, the correction candidates for a are latent, only affecting the values of the feature functions.[9] The final end-to-end system uses the following 11 features:

- 7 error correction model features at the phrase level
- IME conversion probability at the phrase level
- language model probability at the sentence level
- word/phrase penalty features at the sentence level

The language model at the sentence level is trained on a large monolingual corpus of Chinese in hanzi, consisting of about 13 million sentences (176 million words). The IME conversion probability

[8] No detail of this data is available in Chen and Lee (2000).

[9] The final phrase table needs to be unique for each phrase pair (a, w), though the process described here results in multiple entries with the same pair having different feature values, because the generation of (a, w) is mediated by multiple correction candidates $c_{1\ldots k}$. These entries need to be added up to remove duplicates; we used a heuristic approximation of taking the pair where a equals c_j (i.e., no spelling correction) when multiple entries are found.

also uses a word trigram model, but it is trained on a different data set which we did not have access to; we therefore used both of these models. The values for *k* and *n* can be determined empirically; we used 20 for both of them.[10] This generates maximally 400 conversion candidates for each input pinyin.

The feature weights of the log-linear model are tuned using MERT. As running MERT on a CER-based target criterion on the similar, monotone translation task of spelling correction did not lead to a significant improvement (Section 3.3), we simply report the results of using the 4-gram BLEU as the training criterion in this task.

4.3 Experiments and results

For the evaluation of the end-to-end conversion task, we used the CHIME corpus mentioned above. In order to use the word trigram language model that is built in-house, we re-segmented the CHIME corpus using our word-breaker, resulting in 12,102 words in 2,000 sentences. We then divided the sentences in the corpus randomly into two halves, and performed a two-fold cross validation evaluation. The development portion of the data is used to tune the weights of the feature functions in MERT-style training. We measured our results using character error rate (CER), which is based on the longest common subsequence match in characters between the reference and the best system output. This is a standard metric used in evaluating IME systems (e.g., Mori et al., 1998; Gao et al., 2002). Let N_{REF} be the number of characters in a reference sentence, N_{SYS} be the character length of a system output, and N_{LCS} be the length of the longest common subsequence between them. Then the character-level recall is defined as N_{LCS}/N_{REF}, and the precision as N_{LCS}/N_{SYS}. The CER based on recall and on precision are then defined as 1 – recall and 1 – precision, respectively. We report the harmonic mean of these values, similarly to the widely used F1-measure.

As our goal is to show the effectiveness of the unified approach, we used simpler methods of integrating pinyin error correction with character conversion to create baselines. The simplest baseline is a pre-processing approach: we use the pinyin error correction model to convert *A* into a single best candidate *C*, and run an IME system on *C*. Another more realistic baseline is the noisy channel integration discussed in Section 4.1. We approximated this integration method by re-ranking all the candidates generated by the proposed log-linear model with only the channel and language model probabilities, equally weighted.

	CER on 1-best	CER on 5-best
Baseline: No correction	10.91	7.76
Baseline: Pre-processing	9.93	6.75
Baseline: Zheng et al. (2011b)	8.90	
Baseline: Noisy channel	7.92	3.93
Proposed: SMT model	7.12	3.63
Oracle	4.08	1.51

Table 3: CER results for the conversion task (%)

The results are shown Table 3. 5-best results as well as the 1-best results are shown, because in an IME application, providing the correct candidate in the candidate list is particularly important even if it is not the best candidate. Let us first discuss the 1-best results. The CER of this test corpus using the in-house IME system without correcting any errors is 10.91. The oracle CER, which is the result of applying the IME on the gold standard pinyin input derived from the reference text using a hanzi-to-pinyin converter (as mentioned in Section 3.3), is 4.08, which is the upper-bound imposed by the IME conversion accuracy. The simple pipeline approach of concatenating the pinyin correction component with the character conversion component improves the CER by 1% to 9.93. Assuming that there are on average 20 words in a sentence, and each word consists of 2 characters, 1% CER reduction means one improvement every 2.5 sentences. Noisy channel integration improves over this quite substantially, achieving a CER of 7.92, demonstrating the power of the word language model in character conversion. Incidentally, the CER of the output by Zheng et al. (2011b)'s model is 8.90.[11] Their results are not as good as our noisy channel integration, as their system uses a manually defined error model and a word bigram language model. With the use of additional feature functions weighted discriminatively for the final conversion task, the

[10] From Table 2, we observe that the accuracy of the 20-best output of the spelling correction component is over 99%. An offline run with the IME system on an independent data set also showed that the accuracy of the 20-best IME output is over 99%.

[11] Available at http://chime.ics.uci.edu/.

Overall errors (words)	1,074 / 12,102
Conversion	646 (60.14%)
Over-corrections	155 (14.43%)
Under-correction	161 (14.99%)
Wrong correction	112 (10.42%)

Table 4: Classification of errors

proposed method outperforms all these baselines to reduce the CER to 7.12, a 35% relative error rate reduction compared with the no correction baseline, a 20% reduction against Zheng et al (2011b) and a 10% reduction from our noisy channel baseline. The 5-best results follow the same trend of steady improvement as we use a more integrated system.

In order to understand the characteristics of the errors and remaining issues, we ran an error analysis on the 1-best results of the proposed system. For each word in the test data (all 2,000 sentences) for which the system output had an error, we classified the reasons of failure into one of the four categories: (1) **character conversion error**: correct pinyin was input to the IME but the conversion failed; (2) **over-correction of pinyin input**: the system corrected the pinyin input when it should not have; (3) **under-correction of pinyin input**: the system did not correct an error in the input pinyin when it should have; (4) **wrong correction**: input pinyin string had a spelling error but it was corrected incorrectly.

Table 4 shows the results of the error analysis. We find that somewhat contrary to our expectation, over-correction of the spelling mistakes was not a conspicuous problem, even though the pinyin correction rate of the training data is much higher than that of the test data. We therefore conclude that the error correction model adapts very well to the characteristics of the test data in our integrated SMT-based approach, which trains the unified feature weights to optimize the end goal.

5 Conclusion and Future Work

In this paper we have presented a unified approach to error-tolerant text input, inspired by the phrase-based SMT framework, and demonstrated its effectiveness over the traditional method based on the noisy channel model. We have also presented a new method of automatically collecting parallel data for spelling correction from user keystroke logs, and showed that the log-linear model works well on the task of spelling correction in isolation as well.

In this study, we isolated the problem of spelling errors and studied the effectiveness of error correction over a basic IME system that does not include advanced features such as abbreviated input (e.g., typing only "py" for 朋友 *pengyou* 'friend' or 拼音 *pinyin* in Chinese) and auto-completion (e.g., typing only "ari" for ありがとう *arigatou* 'thank you' in Japanese). Integrating data-driven error correction feature with these advanced features for the benefit of users is the challenge we face in the next step.

Acknowledgements

We are indebted to many colleagues at Microsoft and MSR for their help in conducting this research, particularly to Xi Chen, Pallavi Choudhury, Chris Quirk, Mei-Yuh Hwang and Kristina Toutanova. We are also grateful for the comments we received from the reviewers of this paper.

References

Baba, Y. and H. Suzuki. 2012. How are spelling errors generated and corrected? A study of corrected and uncorrected spelling errors using keystroke logs. In *Proceedings of ACL*.

Banko, M. and E. Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of ACL*.

Bertoldi, N., M. Cettolo, and M. Federico. 2010. Statistical machine translation of texts with misspelled words. In *Proceedings of HLT-NAACL*.

Brill, E., and R. C. Moore. 2000. An improved error model for noisy channel spelling correction. In *Proceedings of ACL*.

Chen, Z., and K. F. Lee. 2000. A new statistical approach to Chinese Pinyin input. In *Proceedings of ACL*.

Cherry, C., and H. Suzuki. 2009. Discriminative substring decoding for transliteration. In *Proceedings of EMNLP*.

Cucerzan, S., and E. Brill. 2004. Spelling correction as an iterative process that exploits the collective knowledge of web users. In *Proceedings of EMNLP*.

Duan, H., and P. Hsu. 2011. Online spelling correction for query completion. In *Proceedings of WWW*.

Gao, J., J. Goodman, M. Li and K.-F. Lee. 2002. Toward a unified approach to statistical language modeling for Chinese. In *ACM Transactions on*

Asian Language Information Processing, Vol. 1, No. 1, pp 3-33.

Gao, J., X. Li, D. Micol, C. Quirk and X. Sun. 2010. A large scale ranker-based system for search query spelling correction. In *Proceedings of COLING*.

Jiampojamarn, S., G. Kondrak and T. Sherif, 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *Proceedings of HLT/NAACL*.

Kernighan, M., K. Church, and W. Gale. 1990. A spelling correction program based on a noisy channel model. In *Proceedings of COLING*.

Koehn, P., F. Och and D. Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*.

McEnery, A. and Xiao, Z. 2004. The Lancaster Corpus of Mandarin Chinese: A Corpus for Monolingual and Contrastive Language Study. In *Proceedings of LREC*.

Mori, S., M. Tsuchiya, O. Yamaji and M. Nagao. 1998. *Kana-kanji conversion by a stochastic model. In Proceedings of Information Processing Society of Japan*, SIG-NL-125-10 (in Japanese).

Och, F. J. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*.

Och, F., and Ney, H. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4): 417-449.

Rama, T., A. K. Singh and S. Kolachina. 2009. Modeling letter-to-phoneme conversion as a phrase based statistical machine translation problem with minimum error rate training. In *Proceedings of the NAACL HLT Student Research Workshop and Doctoral Consortium*.

Sherif, T. and G. Kondrak. 2007. Substring-based transliteration. In *Proceedings of ACL*.

Sun, X., J. Gao, D. Micol and C. Quirk. 2010. Learning phrase-based spelling error models from clickthrough data. In *Proceedings of ACL*.

Suzuki, H. and J. Gao. 2005. A comparative study on language model adaptation using new evaluation metrics. In *Proceedings of EMNLP*.

Toutanova, K., and R. C. Moore. 2002. Pronunciation modeling for improved spelling correction. In *Proceedings of ACL*.

Tokunaga, H., D. Okanohara and S. Mori. 2011. Discriminative method for Japanese kana-kanji input method. In *Proceedings of the Workshop on Advances in Text Input Methods* (WTIM 2011).

Whitelaw, C., B. Hutchinson, G. Y. Chung, and G. Ellis. 2009. Using the web for language independent spellchecking and autocorrection. In *Proceedings of ACL*.

Zhang, Y., L. Deng, X. He and A. Acero. 2011. A novel decision function and the associated decision-feedback learning for speech translation. In *Proceedings of ICASSP*.

Zheng, Y., L. Xie, Z. Liu, M. Sun. Y. Zhang and L. Ru. 2011a. Why press backspace? Understanding user input behaviors in Chinese pinyin input method. In *Proceedings of ACL*.

Zheng, Y., C. Li and M. Sun. 2011b. CHIME: An efficient error-tolerant Chinese pinyin input method. In *Proceedings of IJCAI*.