Correcting Chinese Spelling Errors with Word Lattice Decoding

YU-MING HSIEH, Academia Sinica and National Tsing Hua University
MING-HONG BAI, SHU-LING HUANG, and KEH-JIANN CHEN, Academia Sinica

Chinese spell checkers are more difficult to develop because of two language features: 1) there are no word boundaries, and a character may function as a word or a word morpheme; and 2) the Chinese character set contains more than ten thousand characters. The former makes it difficult for a spell checker to detect spelling errors, and the latter makes it difficult for a spell checker to construct error models. We develop a word lattice decoding model for a Chinese spell checker that addresses these difficulties. The model performs word segmentation and error correction simultaneously, thereby solving the word boundary problem. The model corrects nonword errors as well as real-word errors. In order to better estimate the error distribution of large character sets for error models, we also propose a methodology to extract spelling error samples automatically from the Google web 1T corpus. Due to the large quantity of data in the Google web 1T corpus, many spelling error samples can be extracted, better reflecting spelling error distributions in the real world. Finally, in order to improve the spell checker for real applications, we produce n-best suggestions for spelling error corrections. We test our proposed approach with the Bakeoff 2013 CSC Datasets; the results show that the proposed methods with the error model significantly outperform the performance of Chinese spell checkers that do not use error models.

Categories and Subject Descriptors: I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Language models; Text analysis*

General Terms: Algorithms, Languages, Experimentation, Performance

Additional Key Words and Phrases: Chinese spelling error checking, computer-assisted language learning, noisy channel model, word lattice, word segmentation, unknown word detection

ACM Reference Format:
Hsieh, Y.-M., Bai, M.-H., Huang, S.-L. and Chen, K.-J. 2015. Correcting Chinese spelling errors with word lattice decoding. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.* 14, 4, Article 18 (October 2015), 23 pages.
DOI: http://dx.doi.org/10.1145/2791389

1. INTRODUCTION

A spell checker is a writing assistance tool which provides users with better word suggestions by automatically detecting spelling errors in documents. For high-quality publications, one important criterion for judging the quality of the document is a low number of typographical errors. Because manual proofreading is costly and time consuming, computer-assisted tools have been used since the early 1960s [Damerau 1964]. However, the true breakthrough was not until the 1990s, when the noisy channel model started to be used to improve spelling detection and correction [Kernighan et al. 1990; Mays et al. 1991]. When using the noisy channel model to check spelling, based on the principle of information theory [Shannon 1948], spelling correction is seen as

Authors' addresses: Y.-M. Hsieh, Department of Computer Science, National Tsing-Hua University, Taiwan; email: morris@iis.sinica.edu.tw; M.-H. Bai, S.-L. Huang and K.-J. Chen, Institute of Information Science, Academia Sinica, Taiwan.

DOI: http://dx.doi.org/10.1145/2791389

the decoding of a noisy channel. The spell checker is defined as follows: given a word w we seek the correct word c that maximizes the probability of c given w:

$$\operatorname*{argmax}_c P(c|w).$$

Using Bayes' theorem, the given formula is equivalent to

$$\operatorname*{argmax}_c \frac{P(w|c) \cdot P(c)}{P(w)}.$$

Since $P(w)$ is the same for each possible c, the formula is further simplified to

$$\operatorname*{argmax}_c P(w|c) \cdot P(c),$$

where $P(c)$, the *language model*, represents the possibility of c appearing in a sentence, and $P(w|c)$, the *error model*, represents the possibility of c being miswritten as w. In their research, Kernighan and Mays demonstrated that combining the language and error models yields better results than either model alone. Because the noisy channel model works so well and is easy to implement, most spell checkers are based on it, with only minor variations.

For Chinese, most spell checkers of the last twenty years have also been based on this noisy channel framework. However, Chinese spell checkers are much more difficult to develop because of certain Chinese language characteristics.

(1) *No word boundaries*. There are no word boundaries in Chinese text, so we must rely on word segmentation systems to identify Chinese words. Furthermore, a Chinese character may function as a word or a word morpheme, further complicating word identification.
(2) *Large character set*. Unlike alphabetical languages such as English and French whose words are composed of a limited set of letters, Chinese words are composed of a large set of more than ten thousand characters.

Difficulties in Chinese spelling correction are caused by these two characteristics, and they have delayed progress in Chinese spell checking research. We will discuss these issues in detail.

1.1. No Word Boundaries

In English, spelling errors can be essentially classified into two types: *nonword errors*, referring to misspelled words which are not valid words, and *real-word errors*, referring to misspelled words which are valid words but ungrammatical or logically improper. Nonword errors in English can be determined by consulting a dictionary. In Chinese, spelling errors can likewise be classified as nonword and real-word errors, but we cannot simply consult dictionaries to differentiate between the two errors, because there are no blanks to mark word boundaries in Chinese.

For example, when the Chinese word 書櫃 (*shu-gui* "bookshelf") is miswritten as 書貴 (*shu gui* "book expensive"), although it is not a word, we still cannot guarantee that 書貴 (*shu gui*) is necessarily a spelling error because in Chinese a single character can be either a word or a morpheme. That is, 書貴 (*shu gui*) could be a misspelling of 書櫃 (*shu-gui* "bookshelf"), or it could be read as the two words 書 (*shu* "book") and 貴 (*gui* "expensive") in a sentence like 讀書貴有新得 (*du-shu gui you xin de* "the precious part of reading is gaining refreshing experience"). Hence in Chinese nonword errors cannot be discovered by simply consulting dictionaries; instead, we must rely on context to detect and discriminate errors.

One may argue that Chinese words could be automatically identified by a Chinese word segmentation system. However, word segmentation works well only on sentences that contain no spelling errors, as it treats unrecognized character sequences as monosyllabic words; this makes it difficult for word-based language models to spot and correct errors. For instance,

我們	家書	貴	倒	了
wo-men ,	*jia-shu* ,	*gui* ,	*dao* ,	*le* .
we	a family letter	expensive	fall	LE

In the given sentence, because 櫃 (*gui* "cabinet") was miswritten as 貴 (*gui* "expensive"), it results in the further error of identifying 家書 (*jia-shu* "family letter") as a word, resulting in misleading context for the word n-gram language model, which then miscalculates the conditional probabilities of 櫃 (*gui* "cabinet") as well as 貴 (*gui* "expensive"). Even worse, multiple spelling errors in a given sentence can lead to cascaded segmentation errors, which are difficult to correct. Some researchers thus use character-based language models [Huang et al. 2007; Hsieh et al. 2013] to avoid these problems. However, word-based language models are much more robust than character-based language models [Gu et al. 1991; Yang et al. 1998; Gao et al. 2002].

In order to design a reliable Chinese spelling error detection system, some researchers have attempted to detect unknown words [Chen and Bai 1998; Chen and Ma 2002] to aid in detecting spelling errors [Huang et al. 2007; Hsieh et al. 2013; Wu et al. 2013; Chiu et al. 2013]. They find that after word segmentation procedures, a Chinese sentence with spelling errors usually contains ungrammatical single characters, such as 措 (*cuo*) and 折 (*zhe*) in the sentence 不怕 措(?) 折(?) 地 奮鬥 (*bu-pa cuo zhe di fen-dou*). They combine every character in the confusion set of each ungrammatical single word with the preceding and following words, and if one of the resultant candidate words is recognized as a new word, then they regard this new compound as a candidate for replaceable words. This method can detect nonword errors and limits the error candidates, but is largely ineffective for real-word errors due to the difficulty of detecting ungrammatical words. For example, in 牽一髮而動全身 (*qian yi-fa er dong quan-shen* "pull one hair and the whole body moves"), when 全身 (*quan-shen* "whole body") is miswritten as 合身 (*he-shen* "fit the body"), it results in a sentence that is not logical but still grammatical.

1.2. Large Character Set

In past studies, English spelling errors have been roughly summarized into four types of editing operations [Damerau 1964]:

Insertion—Spelling error with redundant letters, for example, *cress* is miswritten as *acress*;

Deletion—Spelling error with lost letters, for example, *actress* is miswritten as *acress*;

Substitution—Spelling error with wrong letters, for example, *access* is miswritten as *acress*;

Transposition—Spelling error with reverse letters, for example, *caress* is miswritten as *acress*.

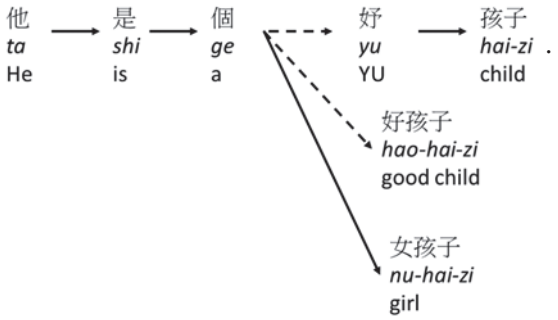
According to Peterson [1986], edit distance 1 covers about 90% of spelling errors, and edit distance 2 accounts for almost all of the remaining spelling errors. Thus the error model probability $p(c|w)$ for English can be simplified to the probability of the edit distance between c and w. For example, $p(access|acress)$ can be simplified to $p(r \rightarrow c)$. This

leaves us with the challenge of establishing within the probability model a confusion matrix of letters by which we may evaluate the probability of each letter representing one of the four types of spelling errors. Since English letters compose a small closed set, a small amount of error pairs are sufficient to yield a good result for the confusion matrix probabilities. However, the Chinese character set is large, consisting of more than ten thousand characters [Unicode Consortium 2014]; the number of commonly used characters is estimated at 5,000 [MOE 1994]. To establish a Chinese 5000-word confusion matrix even considering only the substitution error type, we would need to calculate substitution probabilities for 5,000 × 5,000 words. This is infeasible in practice. Furthermore, in Chinese, insertion, deletion, and transposition errors are considered grammatical errors, since all Chinese characters may also be words; hence inserted, deleted, or transposed characters are merely grammatical errors. Thus Chinese spelling checkers consider character substitution errors only.

To handle substitution errors, some researchers have found that most Chinese spelling errors are caused by characters with similar pronunciations, shapes, or meanings [Chang 1995; Liu et al. 2011; Chen et al. 2011]. Therefore a confusion character set is created using similar characters for each character, based on the previous similarity features. For example, the confusion set of 奈 (*nai*) may include the similarly-shaped characters {捺 (*na*), 柰 (*nai*), 禁 (*jin*), . . . } and similarly-pronounced characters {耐 (*nai*), 鼐 (*nai*), 賴 (*lai*), . . . }. Such a confusion set suggests a limited number of candidates for correction and greatly reduces the parameters that must be estimated by the spelling checker's language model for use in decoding. Therefore the first design issue for Chinese spelling check is how to assemble a good confusion set for each character which is not only small in size but also boasts a high inclusion rate for correct spelling candidates. The second design issue is how to develop a model for spelling check which best estimates the probability of the correct output character sequence for an input sentence; this however calls for a large training corpus of spelling errors. In other words, to estimate the confusion set parameters for the error model, the large character set of Chinese necessitates a bigger training dataset of spelling errors than is needed for English. To avoid the difficulties of constructing Chinese error models, many researchers adopt only a language model when estimating the correctness of word substitution [Chang 1995; Lin et al. 2002; Huang et al. 2007; Hsieh et al. 2013; Jia et al. 2013], resulting in more false alarms. Consider the following examples.

Because the probabilities of 知己 (*zhi-ji* "bosom friend") and 一半 (*yi-ban* "half") are lower than that of the corresponding 自己 (*zi-ji* "myself") and 一般 (*yi-ban* "generally"), the former are substituted by the latter. Thus if a conventional language model is

using in decoding, it always chooses the word sequence with the highest probability. However, the objective of the spelling checker is to find the word sequence that most likely expresses the intended meaning given the input expression, such as



Because the language model probability of p(他是個女孩子) (*ta shi ge nu-hai-zi* "he is a girl") is much greater than p(他是個好孩子) (*ta shi ge hao-hai-zi* "he is a good child"), the system favors 女孩子(*nu-hai-zi* "girl") over 好孩子(*hao-hai-zi* "good child"). However the 妤(*yu*) and 好(*hao*) characters are much more similar than 妤(*yu*) and 女(*nu*). Therefore better spelling error decoding can be achieved by combining the conventional language model with an error model.

In this article we propose a *word lattice decoding* model for Chinese spell checking that solves these problems. The model corrects nonword errors as well as real-word errors. In addition, for better probability estimation of error models, we also propose a methodology for extracting spelling error samples automatically from the Google web 1T corpus. Due to the large quantity of data in the Google web 1T corpus, a sufficient number of spelling error samples can be extracted that reflect real-world spelling error distributions. Finally, in order to make the spelling checker more practical, we introduce a methodology for providing n-best suggestions for the user to choose from.

This article is divided into seven sections. In Section 2 related works are reviewed and addressed, and in Section 3 we introduce and discuss the proposed system. The extraction of spelling error samples is described in Section 4, and the results are presented in Section 5. In Section 6, we discuss extending the model to handle other types of errors, and in Section 7 we conclude and describe directions for future work.

2. RELATED WORKS

In the early 1990s, Kernighan et al. [1990] and Mays et al. [1991] first applied the noisy channel model to spell checkers. Due to the framework's simplicity and effectiveness, most later studies also adopted and improved upon this model. The Chinese spell checking system is basically also based on the noisy channel model. However, due to the special characteristics of the Chinese language, as mentioned in Section 1, a Chinese spell checker is more difficult to design than an English one. We will address these challenges and discuss the proposed solutions.

Regarding Chinese error detection, researchers [Ren et al. 2001; Huang et al. 2007; Chen et al. 2011; Hsieh et al. 2013] found that similar to unknown words, Chinese spelling errors usually result from ungrammatical single characters after word segmentation. They thus likewise detected unknown words [Chen and Bai 1998; Chen and Ma 2002] to detect spelling errors. That is, they used ungrammatical single characters as clues to look for spelling errors. They found this approach to be simple and effective, although among the two types of spelling errors, most real-word errors cannot be detected in this way. Because ungrammatical words are difficult to detect, an

ungrammatical character can be detected by boundedness of the character and by using simple grammatical patterns. For example, when 全身 (*quan-shen* "whole body") is miswritten as 合身 (*he-shen* "fit the body"), because the latter is also a recognizable word, the unknown-word detection method does not detect spelling errors in such a real-word example.

Many researchers have used error template rules to detecting spelling errors [Huang et al. 2008; Chen et al. 2009, 2011]. An error template rule usually is formed by tuple $<X_1, X_2>$, where the first column X_1 denotes the string that contains spelling errors, and the second column X_2 denotes the string that has been corrected, for example, <不同凡想 (*bu-tong-fan-xiang*), 不同凡響(*bu-tong-fan-xiang* "extremely good")>. If X_1 appears in a sentence, the spell checker then marks it as a possible error. Since error template rules are usually derived from a spelling error corpus with many contexts, the precision of error detection when using error template rules is relatively high. On the other hand, as the recall rate of error template rules is very low, this approach is more suitable during pre- or post-processing to detect common spelling errors.

As proposed by Chang et al. [1995], another way to improve error detection is to treat every character as a potential typo and provide candidate characters for each error. Through noisy channel model calculations, the original word competes with its candidate words; if the score of the original word is lower than its candidate words, the original word is regarded as a typo and is replaced by the candidate with the highest score. This method, which considers every word as a potential spelling error, is similar to our approach. However, they use only a language model, resulting in excessive false alarms, since language models always select the words or characters with the highest probability.

As mentioned in Section 1, in error model estimation, the Chinese character set is large. There are about 5,000 commonly used characters, so it is unlikely that each character can be treated as a potential candidate, as is done in English. For this reason, most Chinese spell-check systems rely on a reasonably-sized confusion set [Zhang et al. 2000; Lin et al. 2002; Liu et al. 2008, 2009a, 2009b, 2011]. Since the confusion set coverage strongly affects the recall rate of a correction system, in order to establish a confusion set with sufficient coverage, researchers have summarized the three major causes of spelling errors (similar pronunciation, shape, or meaning) as the criteria upon which to build a confusion set. However, among them, generally only similar-shape errors are addressed. For instance, Zhang et al. [2000] generated a confusion set by calculating the encoding similarity of the five stroke input method; and Lin et al. [2002] generated a confusion set by calculating the Cangjie code similarity. Later, Liu et al. [2008] found that using this approach is flawed, because Cangjie is specifically designed for input and sets a five-code limit for each word; hence more complicated character shapes are simplified to meet the code limits – but such simplification is not conducive to character shape comparison. They thus proposed a methodology to extend Cangjie code to achieve more precise shape comparisons.

Similar-shape studies have contributed to Chinese confusion set generation, but quantifying the similarity of character shapes for error modeling is another issue. Some researchers simply ignore the error model for lack of reliable similarity measures, and instead rely on the language model alone to select the correct word [Chang et al. 1995; Ren et al. 2001; Huang 2002, 2007; Jia et al. 2013; Hsieh et al. 2013]. However, as mentioned earlier, if the similarity of the error model is ignored, the language model favors more probable word sequences over more reasonable words, leading to incorrect spelling. Chen et al. [2011] attempted training the error model probabilities on a corpus of student writing with spelling correction. Although this approach leveraged real spelling errors, such manually-edited data is difficult to acquire. Liu et al. [2009a,

2009b, 2011] presented three criteria based on the assessment of the difference in the Cangjie codes. Chiu et al. [2013] also proposed a simple frequency ratio between the error word and the correction as the score of the error model.

Regarding language models, character-based language models and word-based language models are generally adopted. Using a character-based language model is attractive because there is no need for word segmentation and because it requires less memory. Although preliminary tests indicated that the word-based language model is much more powerful than the character-based language model [Gu et al. 1991; Yang et al. 1998; Gao et al. 2002], word-based language models not only suffer from segmentation errors, but also require a corpus of sufficient size to avoid data sparseness. To reduce data sparseness in the language model, Chang [1995] applied the inter-word character bigram, proposed by Lee et al. [1993b], into his language model. The language model is a transformation of a word-class-based language model which represented the class of a word by its suffix and prefix words. The bigram statistics are built based on the co-occurrences of the word suffix of a word and the word prefix of the following word.

Unlike the approach of detecting spelling errors after word segmentation, Jia et al. [2013] propose a graph model; they simultaneously segment sentences and detect spelling errors. This is very similar to our approach of word lattice decoding, except that while scoring the decoding paths, they employ only the language model and ignore the error model.

In addition to research on Chinese spell checking, in automatic speech recognition the noisy channel model has also been adopted to find the best character sequence for a sequence of acoustical signals. Lee et al. [1993a, 1993b] propose a Chinese speech recognition model that is very similar to our word lattice decoding model. However, the estimation of the acoustic model in automatic speech recognition is quite different from that for the error model of a spell checker. This is because the error model estimates the probabilities of a miswritten character to other characters with similar pronounciations, shapes, or meaning, as mentioned in Section 1.

3. THE SPELL CHECKER

Currently, most Chinese spell checkers first perform word segmentation, and then carry out detection and correction based on the results of segmentation. The disadvantage of this approach is that spelling errors often affect the word segmentation results, which then leads to additional errors during the detection and correction processes. Therefore with our spell checker we adopt a word lattice decoding approach in which we simultaneously include all confusing characters for each character of an input sentence, from which we create the word lattice. This results in word lattice nodes that carry not only the words from original sentence but also the potential substitution words contained in the confusion sets. Consider the example of 不怕措折地 (*bu-pa cuo zhe di*), as shown in Figure 1; substituting the confusion word 挫 (*cuo*) for 措 (*cuo*) yields a recognizable compound 挫折 (*cuo-zhe* "frustration"), so we add a node for it in the word lattice. In other words, by finding the best path in the resultant word lattice we find not only the best word segmentation result but also the best substitute words. For a more realistic example, see Section 3.1.

Given a sentence which may contain spelling errors $S = s_1,s_2,\ldots,s_n$, our correction procedure can be divided into two major steps.

(1) Create the word lattice for $\{s_1,s_2,\ldots,s_n\}$ and their confusion characters: For each character in $\{s_1,s_2,\ldots,s_n\}$ find its confusion characters and extend every character in S into a character candidate list. Consult a dictionary D: if the words preceding

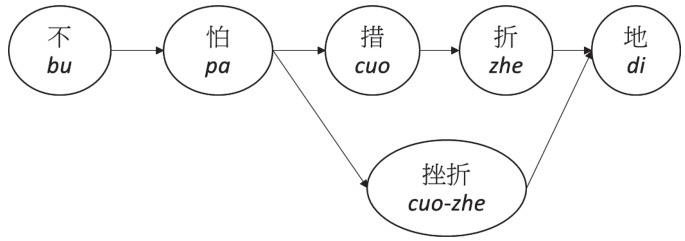


Fig. 1. Building a word lattice with a character confusion set.

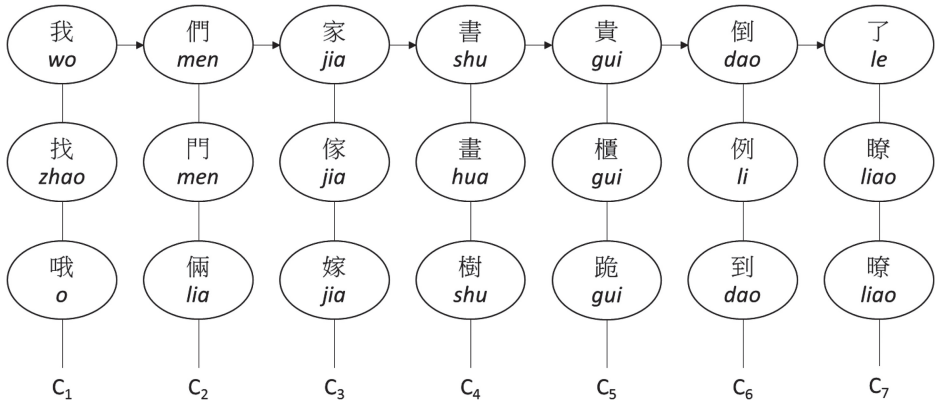


Fig. 2. Each character's set of confusion characters.

or following the character candidate list can be combined to form a longer compound word, then add the compound word to the word lattice as a new node.

(2) According to the noisy channel decoding model, decode the word lattice to find the best path.

After word lattice decoding we have a best path $\widehat{W}$, which is still an n-character sentence. However, unlike the original sentence S, for each character c_i of $\widehat{W}$, we have either $c_i = s_i$ or $c_i \neq s_i$: $c_i = s_i$ if s_i has been identified by the system as the correct character, and $c_i \neq s_i$ if s_i has been identified as an incorrect character, the suggested replacement for which is c_i.

We describe in Section 3.1 how to construct the word lattice from sentence S by step (1), and in Section 3.2 we address step (2) in detail. Finally, in Section 3.3, we demonstrate how to obtain the corrected result from the n-best suggestions.

3.1. Word Lattice Generation

In the first step, we consider each character in the sentence $S = s_1, s_2, \ldots, s_n$ as a potential typo, and select possible substitute characters for each s_{i} from its confusion set. Thus, for each character s_i in S, we have a set of candidates $\mathcal{C}_i$, where $\mathcal{C}_i = \{s_i\} \cup \{c_i | c_i \in \text{confusion set}(s_i)\}$. Take the sentence of 我們家書貴倒了 (*wo-men jia shu gui dao le*) as an example: in Figure 2, listed below each character is its list of confusion characters.

We then consult a dictionary D to find potential words in the neighboring candidate list, as shown in Figure 3. If we find in the neighboring candidate list from $\mathcal{C}_i$ to $\mathcal{C}_j$ a continuous string $\bar{c}_i^j = \mathrm{c}_i, \mathrm{c}_{i+1}, \ldots, \mathrm{c}_j, \mathrm{c}_k \in \mathcal{C}_k$ that is a word entry in the dictionary,

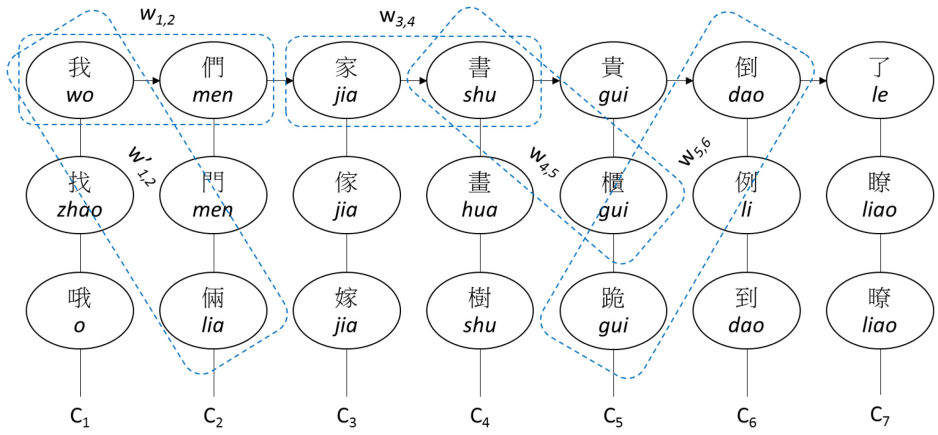


Fig. 3. Potential words identified in neighboring candidate lists.

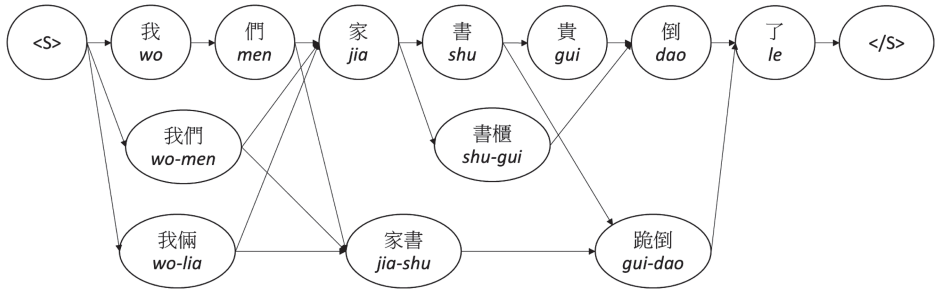


Fig. 4. Word lattice example.

then we generate a word lattice node $w_{i,j} = \bar{c}_i^j$. For example, in Figure 3 the words 我們 (*wo-men* "we"), 我倆 (*wo-lia* "the two of us"), 家書 (*jia-shu* "family letter"), 書櫃 (*shu-gui* "bookshelf"), and 跪倒 (*gui-dao* "kneel down") are added to the word lattice as nodes.

In addition to multisyllabic words, monosyllabic words are also be added to the word lattice, that is, every word s_i in S is added to the word lattice as a new node. However, in order to reduce decoding complexity, only high-frequency monosyllabic words in the confusion set are added into the word lattice.

Then according to each node's position in the sentence, we link the nodes in series. That is, for any node $w_{i,j}$, if there is another node $w_{j+1,k}$ appearing in the j+1-th position, then we link $w_{i,j}$ to $w_{j+1,k}$. Finally, the starting node $w_{0,0}$ and the ending node $w_{n+1,n+1}$ are added to the word lattice as well, and are linked to $w_{1,k}$ and $w_{n,l}$ individually. Word lattice G is defined as G = (V,E), $\mathrm{V} = \{w_{i,j} \,|\, w_{i,j} = c_i, c_{i+1}, \ldots, c_j, w_{i,j} \in D, c_k \in \mathcal{C}_k\} \cup \{w_{0,0}, w_{n+1,n+1}\}$, $\mathrm{E} = \{< w_{i,j} \rightarrow w_{j+1,k} > \,|\, w_{i,j} \in \mathrm{V}, w_{j+1,k} \in \mathrm{V}\}$; an example is shown in Figure 4. Because there are so many nodes with single character words, we omit most of these.

3.2. Word Lattice Decoding

The purpose of word lattice decoding is to find the best path through the word lattice. It is supposed that this path will include both the best segmentation result

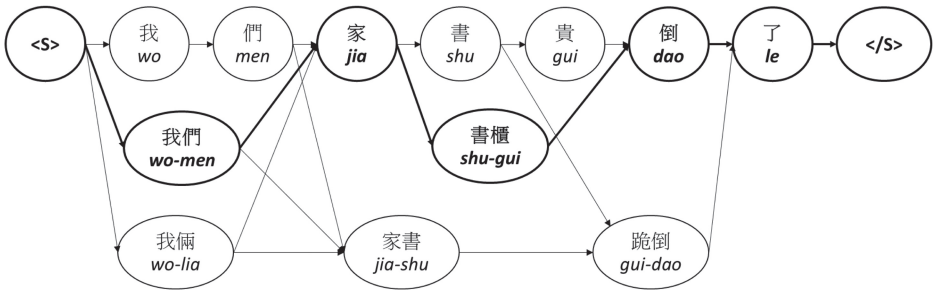


Fig. 5. The best path through a lattice.

and the best substitution words. Expressed mathematically, we seek a path $\widehat{W}$ in the word lattice:

$$\widehat{W} = \underset{W \in \mathcal{L}(S)}{\operatorname{argmax}} P\big(S|W\big)^{\lambda} P(W)^{1-\lambda},$$

where $\mathcal{L}(S)$ is the set of all possible paths through S's word lattice. $P(W)$ is the prior probability of path W as estimated by the n-gram language model. Thus the language model is a quantitative measure of the likelihood that W is a Chinese sentence. $P(S|W)$ is the error model, which measures the likelihood that the author typed S by mistake when W was intended. λ is a weight parameter between the error and language models; the optimal value may be tuned on the development set.

Figure 5 shows a word lattice and the best path $\widehat{W}$ = 我們,家,書櫃,倒,了, (*wo-men jia shu-gui dao le* "our bookshelf fell over") which includes both the segmentation and substituted words, such 櫃 (*gui*) replacing 貴 (*gui*).

For the language model, we calculate the sentence probability using a word-based n-gram language model. For the error model $P(S|W)$, we calculate the sentence probability as the product of the conditional probabilities of words $w_{i,j}$ being written as $\bar{s}_i^j$, defined as

$$\mathrm{P}\left(S|W\right) = \prod_{w_{i,j} \in W} P(\bar{s}_i^j | w_{i,j}).$$

Because $w_{i,j} = c_i, c_{i+1}, \ldots, c_j, c_k \in \mathcal{C}_k$, we further simplify the probability of each character c_k being written as s_k.

$$P(\bar{s}_i^j | w_{i,j}) = \prod_{k=i..j} P(s_k|c_k).$$

The probability of each character c_k being miswritten as s_k is defined as

$$P(s_k|c_k) = \begin{cases} 1 - p_{err}, & s_k = c_k \\ \frac{freq(c_k \to s_k)}{\sum_{\forall t \in \mathcal{C}_k} freq(c_k \to t)} * p_{err}, & s_k \neq c_k, \end{cases}$$

where p_{err} is a control parameter ranging from 0 to 1 which indicates the probability of any character being miswritten. With smaller values of p_{err}, the typo penalty is larger, and the model returns fewer typos. The value of $freq(c_k \to s_k)$ and $freq(c_k \to t)$ can be learned from a spelling error corpus. In our experiments, as described in Section 4, we extracted the spelling error samples from Google Web 1T automatically.

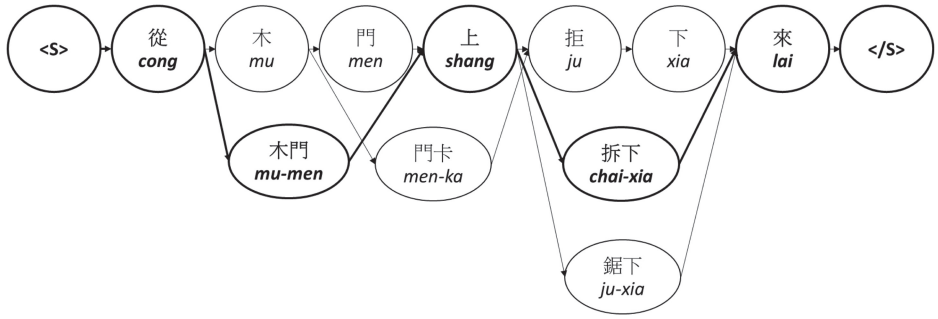


Fig. 6. Computing the score when 拒 (*ju*) is replaced by 鋸 (*ju*).

For the word lattice decoding, we adopt dynamic programming; that is, for any node $w_{i,j}$, we define the forward decoding algorithm recursively as

$$\mathrm{Q}\left(w_{i,j}\right) = \max_{w_{k,i-1}} \left(\mathrm{Q}\left(w_{k,i-1}\right) + \mathrm{q}\left(w_{i,j}\right)\right),$$

$$\begin{aligned} q(w_{i,j}) &= \log\left(P\left(\bar{s}_i^j | w_{i,j}\right)^{\lambda} P\left(w_{i,j} | w_{k,i-1}\right)^{1-\lambda}\right) \\ &= \lambda log P\left(\bar{s}_i^j | w_{i,j}\right) + (1-\lambda) log P\left(w_{i,j} | w_{k,i-1}\right). \end{aligned}$$

Function $\mathrm{Q}\left(w_{i,j}\right)$ denotes the forward decoding probability of node $w_{i,j}$, and $\mathrm{q}(w_{i,j})$ is the combined score of the error and language models. We define the initial value $\mathrm{Q}(w_{0,0}) = 0$. To facilitate calculations, we use log probabilities to calculate the forward decoding probability. When calculating the forward decoding algorithm, the forward decoding probability from $w_{0,0}$ to each node $w_{i,j}$ is calculated along with the best path. Hence when the algorithm has processed the complete word lattice, the best path as well as the log probability are computed.

3.3. N-best Suggestions

In the previous section, we used forward decoding to calculate the best path, obtaining the best error-corrected suggestion for the sentence. We provided for each typo one corrected suggestion; however, in real-world applications, to increase the likelihood of getting the correct answer, users may desire more than one such suggested correction. Taking Figure 6 as an example, forward decoding suggests only the best alternative character 拆 (*ju*) for 拒 (*ju*), but sometimes more suggestions, such as 鋸 (*ju*), would be helpful. According to the definition, the forward decoding algorithm computes $\mathrm{Q}(\mathrm{w}_{ij})$, the best path from the start <S>to any node, say 鋸 (*ju*). But forward decoding does not compute the best path from 鋸 (*ju*) to </S>. To provide n-best suggestions for a given character, we must calculate the best path from the alternative nodes to </S>.

To calculate the best path from any node w_{ij} to </S>, we just reverse the decoding direction from </S>to <S>. We define the backward decoding algorithm as

$$\mathrm{R}\left(w_{i,j}\right) = \max_{w_{j+1,l}} \left(\mathrm{R}\left(w_{j+1,l}\right) + \mathrm{r}(w_{i,j})\right),$$

$$\begin{aligned} \mathrm{r}(w_{(i,j)}) &= \log\left(P\left(\bar{s}_i^j | w_{i,j}\right)^{\lambda} P\left(w_{j+1,l} | w_{i,j}\right)^{1-\lambda}\right) \\ &= \lambda log P\left(\bar{s}_i^j | w_{i,j}\right) + (1-\lambda) log P\left(w_{j+1,l} | w_{i,j}\right). \end{aligned}$$

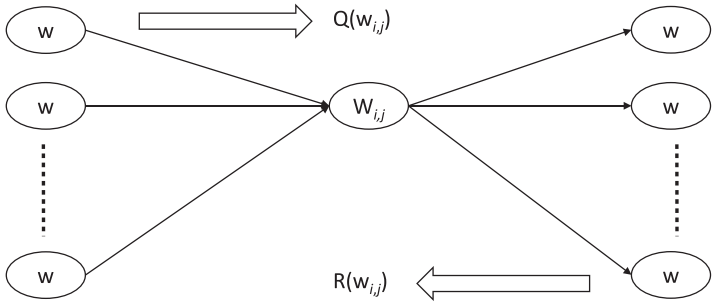


Fig. 7. Forward and backward decoding.

The purpose of backward decoding is to search for the best partial path from node $w_{i,j}$ to node $w_{n+1,n+1}$. Thus, as long as we have performed both forward and backward decoding, for any node $w_{i,j}$ in the word lattice, as shown in Figure 7, we have the best partial path from node $w_{0,0}$ to node $w_{i,j}$ as well as the best partial path from $w_{i,j}$ to $w_{n+1,n+1}$.

Then, for any node $w_{i,j}$ we obtain the best path from $w_{0,0}$ through $w_{i,j}$ to $w_{n+1,n+1}$ by connecting the partial paths from the two algorithms. The decoding probability of the path is then $\mathrm{Q}(w_{i,j}) + \mathrm{R}(w_{i,j})$. Given the probability of any node in the lattice, the score of any character c_k is estimated as the maximum probability of all nodes $w_{i,j}$ that contain the character:

$$\text{score}\,(c_k) = \max_{\forall w_{i,j} \ni C_k} \left(Q\left(w_{i,j}\right) + R(w_{i,j})\right).$$

Finally, we sort the possible candidate characters in each position of the sentence by their scores and then return the n-best suggestions.

4. TRAINING CHARACTER SIMILARITY FROM AN UNTAGGED CORPUS

Web texts are more casual and contain more errors than published books or professional articles because the authors spend less time proofreading the material. Although for much research these errors are seen as noise or as obstacles to be removed, for research on spell checkers, these spelling errors are valuable information. By extracting spelling errors from a large web corpus automatically, we gain not only a sufficient number of spelling error samples but also the actual statistical distribution of spelling errors. In this section, we propose a method of automatically extracting spelling error data from the Google web 1T corpus. The approach can be divided into the following steps.

(1) *Error candidate collection*. Find each string t that contains a spelling error among the n-grams in the corpus, and each corresponding correct word t'.
(2) *Context vector extraction*. For each $<t,t'>$ error candidate pair, collect from the corpus the context vectors $\vec{v_t}$ and $\vec{v_{t'}}$ of t and t'.
(3) *Cosine similarity computation*. Calculate the cosine similarity of each $<t,t'>$ and delete pairs with low similarities.
(4) *Character similarity score computation*. Use the $<t,t'>$ pairs to calculate the spelling error probabilities.

4.1. Error Candidate Collection

The Chinese Google web 1T corpus contains mainly Chinese word n-grams and their frequency counts. These n-grams ranges from unigrams (single tokens) to 5-grams

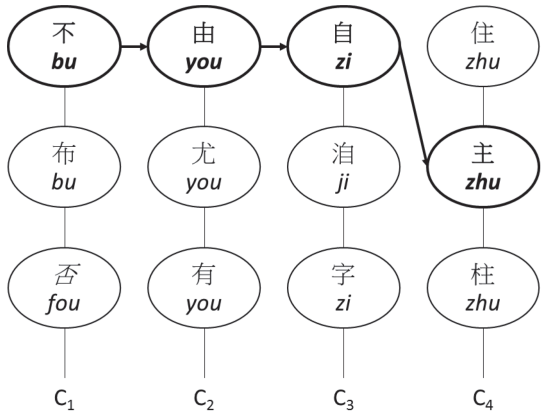


Fig. 8. Example of correcting n-gram 不由自住 (*bu-you-zi-zhu*).

<不由自住 (*bu-you-zi-zhu*), 不由自主 (*bu-you-zi-zhu* “involuntarily”)>
<主意力 (*zhu-yi-li*), 注意力 (*zhu-yi-li* “attention”)>
<乒乒球 (*ping-ping-qiu*), 乒乓球 (*ping-pang-qiu* “table tennis”)>
<乾固 (*gan-gu*), 乾涸 (*gan-he* “to dry up”)>
<人革 (*ren-ge*), 人格 (*ren-ge* “personality”)>
<大陽 (*da-yang*), 太陽 (*tai-yang* “the sun”)>
<上伐 (*shang-fa*), 上代 (*shang-dai* “the previous generation”)>
<絃月 (*xian-yue*), 弦月 (*xian-yue* “crescent moon”) >
<大睦 (*da-mu*), 大陸 (*da-lu* “continent”)>
<讚聲 (*zan-sheng*), 顫聲 (*chan-sheng* “trembling voice”)>

Fig. 9. Error candidate table.

(five-token sequences). Our extraction strategy is to identify each n-gram containing a potential incorrectly spelled word, and then to determine the corrected n-gram. For any term t among the n-grams, if t is not included in the dictionary D, then t may be either a pure n-gram or a term with miswritten characters. We assume that t contains one spelling error: we test each character in t one-by-one by substituting each with the characters in its corresponding confusion set. If we find a new string $t' \in D$, we then add the pair $<t, t'>$into the error candidate table. Note that, when a term contains miswritten characters, the Chinese segmentation system segments the term into separated characters. For example, when 觸目驚心 (*chu-mu-jing-xin* “dreadful”) is miswritten as 觸目警心 (*chu-mu-jing-xin*), it is instead segmented into the three separate tokens “觸目”, “警”, and “心”, and counted as a 3-gram. This shows why we collect error candidates from n-grams.

Take 不由自住 (*bu-you-zi-zhu*) as an example (Figure 8); when 住 (*zhu*) is replaced by 主 (*zhu*), 不由自主 (*bu-you-zi-zhu* “involuntarily”) turns out to be a word in the dictionary, and we add <不由自住 (*bu-you-zi-zhu*), 不由自主 (*bu-you-zi-zhu)*> to the error candidate table. Figure 9 shows sample error candidates; note the words included that are not spelling errors, such as 人革 (*ren-ge*), 上伐 (*shang-fa*), and 讚聲 (*zan-sheng*), which call for further evaluation.

4.2. Similarity Evaluation

In the previous section, we ended up with a large set of error candidate pairs, but many of them were not really spelling errors, and thus require further validation. Our

不由自住 (*bu-you-zi-zhu*) {總是 (*zong-shi* "always"), 身體 (*shen-ti* "body"), 還是 (*hai-shi* "still"), 開始 (*kai-shi* "to start"), 目光 (*mu-guang* "sight"), 身子 (*shen-zi* "body"), 心中 (*xin-zhong* "in the heart"), 退後 (*tui-hou* "back"), 喊出 (*han-chu* "yell"), 我 (*wo* "I"), ...}

不由自主(*bu-you-zi-zhu*) {還是 (*hai-shi* "still"), 身體 (*shen-ti* "body"), 總是 (*zong-shi* "always"), 身子 (*shen-zi* "body"), 眼淚 (*yan-lei* "tears"), 總會 (*zong-hui*), 想起 (*xiang-qi* "to remember"), 發出 (*fa-chu* "to issue"), 想到 (*xiang-dao* "to think of"),竟 (*jing*),...}

Fig. 10. Contextual words for 不由自住 (*bu-you-zi-zhu*) and 不由自主 (*bu-you-zi-zhu* "involuntarily").

Table I. Error Candidate Pairs and Their Cosine Similarities

<不由自住 (*bu-you-zi-zhu*), 不由自主 (*bu-you-zi-zhu* "involuntarily")>	0.0220
<主意力 (*zhu-yi-li*), 注意力 (*zhu-yi-li* "attention")>	0.0392
<乒乒球 (*ping-ping-qiu*), 乒乓球 (*ping-pang-qiu* "table-tennis")>	0.0727
<乾固 (*gan-gu*), 乾涸 (*gan-he* "to dry up")>	0.0001
<人革 (*ren-ge*), 人格 (*ren-ge* "personality")>	6.286×10^{-8}
<大陽 (*da-yang*), 太陽 (*tai-yang* "the sun")>	0.0031
<上伐 (*shang-fa*), 上代 (*shang-dai* "the previous generation")>	7.693×10^{-10}
<絃月 (*xian-yue*), 弦月 (*xian-yue* "a crescent moon") >	0.0004
<大睦 (*da-mu*), 大陸 (*da-lu* "continent")>	2.295×10^{-8}
<讚聲 (*zan-sheng*), 顫聲 (*chan-sheng* "quavery")>	3.278×10^{-7}

assessment approach is based on the assumption that words with similar meanings are often used in similar contexts [Schütze 1999]. We calculate the context similarity of t and t' in texts; a high context similarity suggests that t and t' are synonymous, and thus that t is likely a misspelled word of t'.

Specifically, for each error candidate pair $<t, t'>$, we create context vectors $\vec{v}_t$ and $\vec{v}_{t'}$ for t and t' by collecting the adjacent words preceding and following t and t' in the Google web 1T corpus, respectively. Figure 10 demonstrates this using 不由自住 (*bu-you-zi-zhu*) and 不由自主 (*bu-you-zi-zhu* "involuntarily") as examples.

Next, we use cosine similarity to calculate the similarity of the context vector pairs. Cosine similarity is defined as

$$\cos\left(\vec{v}_t, \vec{v}_{t'}\right) = \frac{\vec{v}_t \cdot \vec{v}_{t'}}{\|\vec{v}_t\| \|\vec{v}_{t'}\|}.$$

Table I shows the example error candidate pairs and their cosine similarities. This shows that the cosine similarity for false-misspelled word pairs are usually very low, suggesting the use of cosine similarity to filter out false drops. Thus filtering the list yields a reliable list of error pairs which can be used for two purposes: for training data in calculating character similarity, and for error template rules [Huang et al. 2008; Chen et al. 2009, 2011] for detecting spelling errors. In the following section, we explain how to calculate character similarity using the error pair list.

4.3. Character Similarity Computation

In the previous section, we extracted a reliable error pair list; we now use the data to calculate character similarity. We estimate the probability of a character c being miswritten as e as

$$p\,(e|c) = \frac{\sum_{\forall(t,t'),t\in e,t'\in c} freq(t)}{\sum_{\forall(t,t'),t'\in c} freq(t)},$$

Table II. Sample Confusion Sets Sorted by Character Similarities

耀 (*yao*)	燿 (*yao*), 0.607, 曜 (*yao*), 0.187, 躍 (*yue*), 0.128, 翠 (*cui*), 0.071, 悅 (*yue*), 0.003, ...
涉 (*she*)	步 (*bu*), 0.842, 社 (*she*), 0.073, 色 (*se*), 0.044, 扯 (*che*), 0.006, 射 (*she*), 0.005, ...
騖 (*wu*)	騖 (*wu*), 0.998, 鵡 (*wu*), 0.001, 兀 (*wu*), $1.79\text{x}10^{-5}$, 霧 (*wu*), $1.30\text{x}10^{-6}$, ...
奈 (*nai*)	捺 (*na*), 0.530, 耐 (*nai*), 0.265, 柰 (*nai*), 0.184, 乃 (*nai*), 0.0143, 奶 (*nai*), 0.003, ...
博 (*bo*)	搏 (*bo*), 0.623, 膊 (*bo*), 0.370, 薄 (*bo*), 0.002, 伯 (*bo*), 0.002, 勃 (*bo*), 0.001, ...
彿 (*fu*)	佛 (*fo*), 0.825, 拂 (*fu*), 0.167, 弗 (*fu*), 0.004, 撫 (*fu*), 0.002, 唬 (*hu*), $1.18\text{x}10^{-4}$, ...
配 (*pei*)	佩 (*pei*), 0.852, 酊 (*ding*), 0.121, 陪 (*pei*), 0.002, 酒 (*jiu*), $8.78\text{x}10^{-4}$, 酷 (*ku*), $4.82\text{x}10^{-5}$, ...
沖 (*chong*)	衝 (*chong*), 0.999, 中 (*zhong*), $3{,}96\text{x}10^{-4}$, 充 (*chong*), $3.27\text{x}10^{-4}$, 茺 (*chong*), $1.84\text{x}10^{-4}$,
	忡 (*chong*) $1.13\text{x}10^{-5}$, ...

where $freq(t)$ denotes the frequency of the n-gram t, which contain spelling errors, appearing in the corpus.

Table II shows actual confusion sets sorted by character similarity. Note that the characters most likely to be miswritten are those that are similar in both shape and pronunciation, for instance, 騖 (*wu*) → 騖 (*wu*), 博 (*bo*) → 搏 (*bo*), and 耀 (*yao*) → 燿 (*yao*); followed by characters with similar shapes, for instance, 涉 (*she*) → 步 (*bu*) and 配 (*pei*) → 酊 (*ding*). Similarly-pronounced characters are more difficult to predict; we found that some were never confused, such as 耀 (*yao*) → 要 (*yao*), and 涉 (*she*) → 設 (*she*). Others, however, are easily miswritten, such as 配 (*pei*) → 佩 (*pei*) and 沖 (*chong*) → 衝 (*chong*). This could be because the latter characters in the pairs are semantically related to the paired characters. Therefore, reliable similarity measures for similarly-pronounced characters can only be estimated by a spelling error corpus.

4.4. Error Template Rules

In order to evaluate the similarities between characters and their confusion set, we proposed a method to extract error candidate pairs. We used a cosine measure of the context vector to evaluate the similarity between the terms with error and correct terms. Error candidate pairs with their cosine similarities can also be used as error template rules in detecting spelling errors. We select highly similar pairs as error template rules. An error candidate rule is defined as

$$< \mathrm{X}_1, \mathrm{X}_2 >,$$

where X_1 denotes a term with a spelling error and X_2 denotes the correction of the term. The application of the rules are very intuitive: we add the terms with spelling errors into the dictionary of a segmentation system, and employ the word segmentation system to identify error terms. Taking rule <不同凡想 (*bu-tong-fan-xiang*), 不同凡響 (*bu-tong-fan-xiang* "extremely good")> as an example, we add 不同凡想 into the segmentation dictionary, and when the system identifies 不同凡想 in a sentence, the system knows that is has detected an spelling error and suggests replacing 想 (*xiang*) with 響 (*xiang*).

他	參加	了	一 場	不同凡想(不同凡響)	的	競賽.
ta	*can-jia*	*le*	*yi chang*	*bu-tong-fan-xiang*	*de*	*jing sai*.
He	enters	LE	an	extraordinary	DE	competition.

Since error template rules are derived from a corpus which includes more contexts, the precision of error detection when using error template rules is relatively high. On

the other hand, the recall rate is very low, so such rules are more suitable for use in pre- or post-processing to detect common spelling errors.

5. EXPERIMENTS

5.1. Experimental Settings

We used three datasets to build our word lattice decoding model. The first one was the Google web 1T corpus; from this we used n-grams from unigrams to 5-grams to extract the miswritten samples, which were further used to train the word similarity list (Sections 4.1–4.3). The samples were also used to extract error template rules (Section 4.4). We used n-grams from unigrams to trigrams to build the language model. At runtime, we used unigrams as a dictionary to create the word lattices for the input sentences. Our second dataset was the CKIP dictionary, which contains about 90,000 Chinese words and is used when extracting the error candidate pairs automatically (Section 4.1). The third dataset was the character confusion set proposed by Liu et al. [2011], which contains 5,401 Chinese common characters and their confusion sets.

For system tuning and testing, we used 700 training sentences from the Bakeoff 2013 CSC Data set [Wu et al. 2013] as our development set, and as our test data we used the test set of subtask 2, which contains 1000 sentences with spelling errors.

5.2. Evaluation Results

In order to identify the advantages and disadvantages between different systems, we used the recall, precision, and F-score metrics to estimate the performance of the various spell checkers. These metrics are defined as

$$recall = \frac{\#\ of\ char\ modified\ correctly}{\#\ of\ error\ chars},$$

$$precision = \frac{\#\ of\ char\ modified\ correctly}{\#\ of\ char\ modified},$$

$$\text{Fscore} = \frac{2 \cdot precision \cdot recall}{precision + recall}.$$

We used the following models in our evaluations.

WLD. This is the proposed word lattice decoding model.

WLD-no-err-model. In order to demonstrate the importance of the error model for the spell checker, we removed the error model of WLD and used only the language model to detect errors and suggest correct substitutes.

WLD+ETR. In order to determine whether or not the error template rule and the WLD are complementary in spelling error detection, we first used the WLD to detect spelling errors, and then applied the error template rules in post-processing. Then we merged the error suggestions from WLD and the error template rules; that is, if a character s_i was detected by the error template rules as an error but was not detected by the WLD model, then we added the suggestion c_i generated by the error template rules to the suggestions of the WLD.

SinicaCKIP. This is the Bakeoff 2013 CSC SinicaCKIP-Run2 system [Hsieh et al. 2013], which uses unknown word detection to detect spelling errors, and uses the Google web 1T language model to suggest the correct substitute.

Table III. Error Correction Evaluation for Various Models

System	Recall	Precision	F-score
WLD-no err model	0.759	0.506	0.607
WLD	0.739	0.764	0.751
WLD+ETR	0.799	0.729	0.762
SinicaCKIP	0.596	0.739	0.660
SinicaCKIP +ETR	0.693	0.737	0.714

Table IV. N-Best Suggestion Coverage

WLD N-best	Coverage
1	0.739
2	0.841
3	0.876
4	0.889
5	0.898
6	0.907
7	0.914
8	0.920
9	0.924
10	0.927

SinicaCKIP+ETR. This is the Bakeoff 2013 CSC SinicaCKIP-Run3 system [Hsieh et al. 2013], which uses the approach described in Section 4 to extract error template rules, coupled with an unknown word detection-based approach to detect spelling errors. The system also uses the Google web 1T language model to suggest the correct substitute. It uses the same merging method as the WLD+ETR model.

From Table III, we observe that the precision rate dropped 25% when using the language model alone without the error model, but the recall rate increased only slightly. This shows that the spell checker performance suffers without the help of the error model. Secondly, the F-score of SinicaCKIP was about 9% lower than that of WLD. WLD's recall and precision were both better than that of SinicaCKIP, in particular for recall. This is consistent with what we mentioned in the first section: unknown word detection-based approaches miss real-word errors. Finally, focusing on error template rules, we find that WLD+ETR outperformed WLD by a mere 1% in terms of the F-score, but outperformed SinicaCKIP by 10%. Thus we conclude that most of the errors detected by the error template rule are also detected by WLD. On the other hand, in terms of precision, the error template rule performance is poorer than that of WLD because fewer errors are detected by the former.

In addition, in Section 3, we proposed a methodology for n-best suggestions; we use the coverage rate to estimate the performance of n-best suggestions. Coverage is defined as

$$coverage = \frac{\#\ of\ char\ covered\ by\ N\ best}{\#\ of\ error\ chars}.$$

From Table IV, we see that the coverage difference between best-1 and best-2 is 10 points, and that between best-2 and best-3 is 3.5 points. This means that the system has difficulty discriminating between the suggestion quality of the top three words. Thus it is important for the spell checker to provide the user with the top three suggestions as references.

Table V. Spelling Error Detection Performances of the Top Systems in the SIGHAN 2013 Subtask1 and the WLD-ETR System

Systems	False-Alarm Rate	Detection Accuracy	Detection Precision	Detection Recall	Detection F-score
Sinica-CKIP+ETR	0.1629	0.842	0.6919	0.8533	0.7642
SJTU	0.0957	0.856	0.769	0.7433	0.7559
SinicaSLMP&NTU	0.1414	0.836	0.7036	0.7833	0.7413
NTHU	0.0514	0.861	0.8455	0.6567	0.7392
KUAS & NTNU	0.2257	0.7890	0.6099	0.8233	0.7007
WLD+ETR	0.1229	0.869	0.7478	0.85	0.7956

5.3. State-of-the-Art Comparison

When comparing our approach with the SIGHAN-2013 Bake-off Chinese Spelling Check Task systems, we use the same metrics [Wu et al. 2013] to evaluate our system. The metrics for the error detection subtask are defined as follows.

- *False-alarm rate* (FAR): # of sentences with false positive errors / # of testing sentences without errors
- *Detection accuracy* (DA): # of sentences with correctly detected results / # of all testing sentences
- *Detection precision* (DP): # of sentences with correctly detected errors / # of sentences reported by the system to have errors
- *Detection recall* (DR): # of sentences with correctly detected errors / # of testing sentences with errors
- *Detection F1* (DF1): 2*DP*DR / (DP+DR)
- *Error location accuracy* (ELA): # of sentences with correct location detection / # of all testing sentences
- *Error location precision* (ELP): # of sentences with correct error locations / # of sentences reported by the system to have errors
- *Error location recall* (ELR): # of sentences with correct error locations / # of testing sentences with errors
- *Error location F1* (ELF1): 2* ELP*ELR / (ELP+ELR)

The metrics for the error correction subtask are defined as follows.

- *Location accuracy* (LA): # of sentences with correct location detection / # of all testing sentences
- *Correction accuracy* (CA): # of sentences with correctly corrected errors / # of all testing sentences
- *Correction precision* (CP): # of sentences with correctly corrected errors / # of sentences for which the system offered corrections

In the spelling error detection task (subtask1), as shown in Table V, Sinica-CKIP had the best performance; our WLD+ETR system further raised the error detection score to 0.79.

For error location detection, as shown in Table VI, our Sinica-CKIP system performance fell to third place, because the error location detection metrics demand that all errors are correctly marked in a sentence when calculating the score; this puts high-recall systems at a disadvantage. However, the proposed WLD-ETR system still achieves the best detection performance.

For the error correction task, as shown in Table VII, Sinica-CKIP made the second place in correction accuracy and third place in correction precision. Our proposed approach made great progress in correction precision.

Table VI. Error Location Detection Performances of the Top Systems in the SIGHAN 2013 Subtask1 and the WLD-ETR System

Systems	Error Location Accuracy	Error Location Precision	Error Location Recall	Error Location F-score
NTHU	0.8200	0.6695	0.5200	0.5854
SJTU	0.8050	0.5931	0.5733	0.5830
Sinica-CKIP+ETR	0.7710	0.5000	0.6167	0.5523
SinicaSLMP&NTU	0.7490	0.4431	0.4933	0.4669
NAIST	0.6980	0.3964	0.5167	0.4486
WLD+ETR	0.812	0.5806	0.66	0.6178

Table VII. Error Correction Performances of the Top Systems in the SIGHAN 2013 Subtask2 and the WLD-ETR System

Systems	Location Accuracy	Correction Accuracy	Correction Precision
NCYU	0.663	0.625	0.703
Sinica-CKIP+ETR	0.559	0.516	0.6158
NTU	0.507	0.467	0.467
NAIST	0.508	0.467	0.577
NTHU	0.454	0.443	0.6998
SinicaSLMP&NTU	0.507	0.467	0.467
WLD+ETR	0.661	0.605	0.629

6. DISCUSSION

As mentioned in Section 1, the difficulty in correcting Chinese spelling errors derives mainly from two Chinese language features: the lack of word boundaries, and the large character set, which contains more than ten thousand characters. These features not only affect substitution errors, but also present great difficulties for insertion, deletion, and transposition errors. In the previous sections, we proposed a noisy channel model to improve the performance of correcting substitution errors by addressing the mentioned difficulties. Recall though that in Chinese, insertions, deletions, and transpositions are considered grammatical errors, since all Chinese characters may also be considered words. However grammatical error correction is out of the scope of this article. We do believe, however, that the proposed model can be extended to correct the other three error types. The crucial considerations when extending the model are (1) How to take different error types into account while generating the word lattice, so that the correct result is one of the alternatives; and (2) In the decoding model, how to adjust the probabilities of the different error types for the model to avoid promoting shorter sentences. For example, the model might consider a modifier of a noun phrase to be an insertion error, since after removing the modifier it yields a valid shorter sentence, resulting in shorter sentences tending to be assigned higher probabilities than the original longer sentences. To avoid this favoring of shorter sentences by the language model, the principle is to multiply the same number of n-gram probability terms for each candidate sentence; that is, in the case of insertion errors, the probability of the candidate sentence should include an additional deletion probability factor, or we should maximize the perplexity of each candidate sentence instead of the generation probability. We present some examples that explain the extension of word lattices and the adjustment of error models.

For insertion errors, take 桌子很有乾淨 (the table is have very clean) as an example, where 有 (*you* "have") is redundant. As shown in Figure 11, to correct this insertion error, we construct a word lattice that includes the empty string ε as a candidate for 有 (*you* "have").

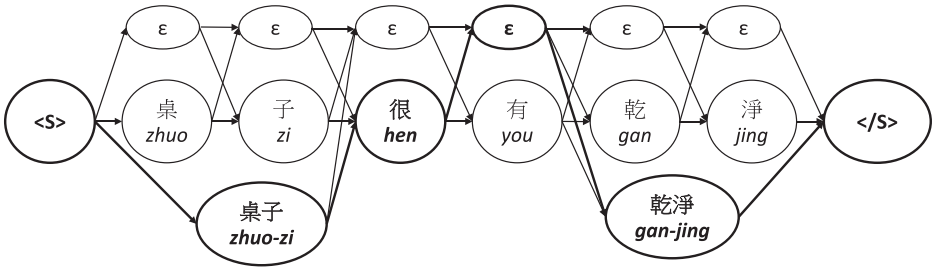


Fig. 11. Word lattice with empty candidate nodes $\varepsilon$for correcting insertion errors.

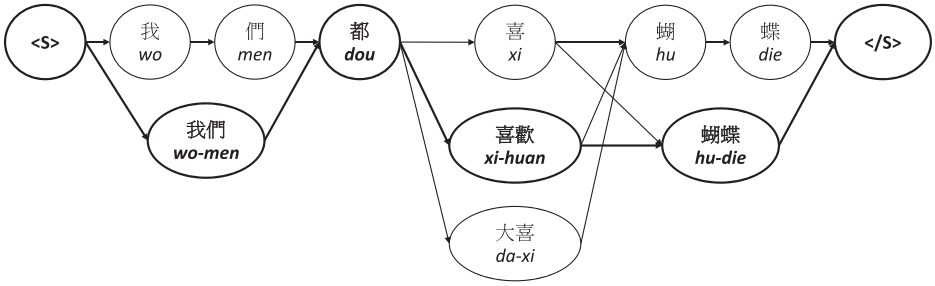


Fig. 12. Word lattice for correcting deletion errors.

In this example, we view every character as a potentially redundant word, where ε nodes indicate the empty string. When the decoder selects an ε node, it is judging the corresponding character to be a redundant word.

For deletion errors, take 我們都喜蝴蝶 (we all joy butterflies) as an example, where 喜歡 (*xi-huan* "like") is miswritten as the single character 喜 (*xi* "joy"). As shown in Figure 12, to resolve this deletion error, we construct a candidate list that includes 歡 (*huan*) when building the word lattice.

In this example, we did not consider the possibility deletion errors occurring for every character, because the large size of the Chinese character set would produce long candidate lists for each character. The most practical method is to take into consideration only bound morphemes when addressing deletion errors. For example, 喜 (*xi*) is rarely used as a single word: this is a clue that helps to detect the deletion error.

Finally, for transposition errors, we take 他義正嚴詞反駁 (he responded sternly) as an example, where 詞嚴 (*ci-yan*) is reversely miswritten as 嚴詞 (*yan-ci*). As Figure 13 shows, we construct a word lattice that includes the correctly-ordered 詞嚴 as a candidate.

In this example, 正義 (*zeng-yi* "justice"), 嚴正 (*yan-zheng* "solemn") and 義正詞嚴 (*yi-zheng-ci-yan* "sternly") are combined into new compounds after correcting the character order, and yield new nodes.

When the word lattice contains every possible candidate for each type of error, the next step is to calculate the error probabilities for the decoding model. For substitution errors, we define the probability of c_k being miswritten as s_k as $\mathrm{p}(s_k|c_k)$ in the error model. To extend the model, we redefine the misspelling probability for each of these three error types respectively. For insertion errors, we redefine the misspelling probability as $\mathrm{p}(s_k|\varepsilon)$, denoting the probability of s_k being mistakenly added to the sentence. For deletion errors, we redefine the probability as $\mathrm{p}(\varepsilon|c_k)$, denoting the probability of the character c_k being mistakenly deleted. For transposition errors, we redefine it as

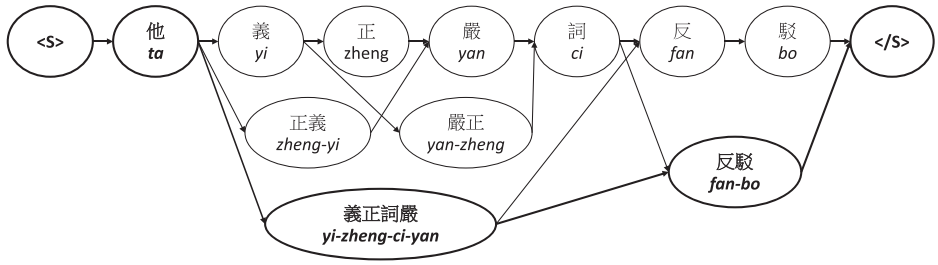


Fig. 13. Word lattice for correcting transposition errors.

$p(s_{k+1}\ s_k|s_k\ s_{k+1})$, denoting the probability of the two characters $s_k\ s_{k+1}$ being written in reverse order. These probabilities can be estimated from a spelling error corpus.

7. CONCLUSIONS

In this article, we proposed using word lattice decoding to correct Chinese spelling errors. By including all the confusion characters in the word lattice, we prevent the word segmentation errors from propagating misspelling-related errors. As to the implementation of the error model, we proposed an innovative method to extract spelling errors from a large web corpus. The extracted spelling error pairs are used both as error template rules and also to estimate the probability of characters in the confusion set. In this study we focused on character substitution errors. We believe that other errors caused by insertions, deletions, and transpositions can also be corrected using this model. In addition, we modified the word lattice decoding of the spell checker to provide n-best suggestions, thus increasing the coverage of correct candidates and allowing users to select the right corrections in real-world systems.

We also found that samples of spelling errors can be extracted automatically from a large web corpus. However there are still many confusion characters that do not occur in the extracted spelling error pairs. Therefore, their similarity measures cannot be derived from spelling error pairs. We thus consider that in the research on error models, Chinese words are broken down into smaller parts, such as by using the Cangjie code; by using a large spelling error corpus we can calculate the error probability for each part of the word, yielding a more accurate error model. For the language model, we here used only the simplest word-based language model. If part-of-speech information were included in the language model, we believe this would improve the selection of the substitute corrections. There are other possible improvements for the current model. Since derivative words like numerals and named entities result in data sparseness in the language model, word-class or interword language models could prove helpful in future work.

Finally, we must address the unknown word problem. According to statistics, Chinese unknown words account for about 5% of text. Since these words are not included in dictionaries and do not appear in corpora, they are easily identified as either unknown or incorrect words. How to prevent them from being regarded as spelling errors is also a challenging research issue.

REFERENCES

Chao-Huang Chang. 1995. A new approach for automatic Chinese spelling correction. In *Proceedings of the Natural Language Processing Pacific Rim Symposium (NLPRS'95)*. 278–283.

Keh-Jiann Chen and Ming-Hong Bai. 1998. Unknown word detection for Chinese by a corpus-based learning method. *Int. J. Comput. Linguistics Chinese Language Process. 3*, 1, 27–44.

Keh-Jiann Chen and Wei-Yun Ma. 2002. Unknown word extraction for Chinese documents. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*, 1–7.

Yong-Zhi Chen, Shih-Hung Wu, Chia-Ching Lu, and Tsun Ku. 2009. Chinese confusion word set for automatic generation of spelling error detecting template. In *Proceedings of the 21st Conference on Computational Linguistics and Speech Processing (ROCLING'09)*. 359–372. [In Chinese]

Yong-Zhi Chen, Shih-Hung Wu, Ping-Che Yang, Tsun Ku, and Gwo-Dong Chen. 2011. Improve the detection of improperly used Chinese characters in students' essays with error model. *Int. J. Continuing Engin. Educ. Life Long Learning 21*, 1, 103–116.

Hsun-Wen Chiu, Jian-Cheng Wu, and Jason S. Chang. 2013. Chinese spelling checker based on statistical machine translation. In *Proceedings of the 7th SIGHAN Workshop on Chinese Language Processing*. 49–53.

Fred J. Damerau. 1964. A Technique for Computer Detection and Correction of Spelling Errors. *Commun. ACM. 7*, 3, 171–176.

Jianfeng Gao, Joshua Goodman, Mingjing Li, and Kai-Fu Lee. 2002. Toward a Unified Approach to Statistical Language Modeling for Chinese. *ACM Trans. Asian Lang. Inform. Process. 1*, 1, 3–33.

Hung-Yan Gu, Chiu-Yu Tseng, and Lin-Shan Lee. 1991. Markov modeling of Mandarin Chinese for decoding the phonetic sequence into Chinese characters. *Computer Speech Lang. 5*, 4, 363–377.

Yu-Ming Hsieh, Ming-Hong Bai, and Keh-Jiann Chen. 2013. Introduction to CKIP Chinese spelling check system for SIGHAN Bakeoff 2013 evaluation. In *Proceedings of the 7th SIGHAN Workshop on Chinese Language Processing*. 59–63.

Chuen-Ming Huang, Mei-Che Wu, and Ching-Che Chang. 2007. Error detection and correction based on Chinese phonemic alphabet in Chinese text. In *Proceedings of the 4th Conference on Modeling Decisions for Artificial Intelligence.* 463–476.

Ta-Hung Hung, Shih-Hung Wu, Tsun Ku, and Wen-Nan Wang. 2008. Chinese essay analysis language model information retrieval. In *Proceedings of the Taiwan E-Learning Forum (TWELF'08)*.

Mark D. Kernighan, Kenneth W. Church, and William A. Gale. 1990. A spelling correction program based on a noisy channel model. In *Proceedings of the 13th International Conference on Computational Linguistics (COLING'90)*. 205–210.

Zhongye Jia, Peilu Wang, and Hai Zhao. 2013. Graph model for Chinese spell checking. In *Proceedings of the 7th SIGHAN Workshop on Chinese Language Processing*. 88–92.

Lin-Shan Lee, Chiu-Yu Tseng, Hung-Yan Gu, F.-H. Liu, C.H. Chang, Y.H. Lin, Yumin Lee, S.L. Tu, S.H. Hsieh, and C.H. Chen. 1993a. Golden Mandarin (I): A real-time Mandarin Speech dictation machine for Chinese language with very large vocabulary. *IEEE Trans. Speech Audio Process. 1*, 2, 158–179.

Lin-Shan Lee, C.-Y. Tseng, and K.-J. Chen, et al. 1993b. Golden Mandarin: An improved single-chip real-time Mandarin dictation machine for Chinese language with very large vocabulary. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'93)*. Vol. 2, 503–506.

Yih-Jeng Lin, Feng-Long Huang, and Ming-Shing Yu. 2002. A Chinese spelling error correction system. In *Proceedings of the 7th Conference on Artificial Intelligence and Applications*.

Chao-Lin Liu and Jen-Hsiang Lin. 2008. Using structural information for identifying similar Chinese characters. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. 93–96.

Chao-Lin Liu, Kan-Wen Tien, Min-Hua Lai, Yi-Hsuan Chuang, and Shih-Hung Wu. 2009a. Phonological and logographic influences on errors in written Chinese words. In *Proceedings of the 7th Workshop on Asian Language Resources*. 84–91.

Chao-Lin Liu, Kan-Wen Tien, Min-Hua Lai, Yi-Hsuan Chuang, and Shih-Hung Wu. 2009b. Capturing errors in written Chinese words. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACLIJCNLP'09)*. 25–28.

Chao-Lin Liu, Min-Hua Lai, Kan-Wen Tien, Yi-Hsuan Chuang, Shih-Hung Wu, and Chia-Ying Lee. 2011. Visually and phonologically similar characters in incorrect Chinese words: Analyses, identification, and applications. *ACM Trans. Asian Lang. Inform. Process. 10*, 2, 1–39.

Eric Mays, Fred J. Damerau, and Robert L. Mercer. 1991. Context based spelling correction. *Inf. Process. Manage. 27*, 5, 517–522.

MOE. 1994. *The Standard Form of National Characters – Instructor's Manual*. Ministry of Education, Taiwan. http://www.edu.tw/files/site_content/M0001/std/c4.htm.

James L. Peterson. 1986. A note on undetected typing errors. *Commun. ACM. 29*, 7, 633–637.

Fuji Ren, Hongchi Shi, and Qiang Zhou. 2001. A hybrid approach to automatic Chinese text checking and error correction. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics.* Vol. 3, 1693–1698.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Comput. Linguistics. 24*, 1, 97–123.

Claude E. Shannon. 1948. A mathematical theory of communication. *Bell Syst. Tech. J. 27*, 3, 379–423.

Unicode Consortium. 2014. The Unicode Standard 7.0. http://www.unicode.org.

Jian-Cheng Wu, Hsun-Wen Chiu, and Jason S. Chang. 2013. Integrating dictionary and web n-grams for Chinese spell checking. *Int. J. Comput. Linguistics Chinese Language Process. 18*, 4, 17–30.

Shih-Hung Wu, Chao-Lin Liu, and Lung-Hao Lee. 2013. Chinese spelling check evaluation at SIGHAN Bake-off 2013. In *Proceeding of the 7th SIGHAN Workshop on Chinese Language Processing (SIGHAN'13)*. 35–42.

Kae-Cherng Yang, Tai-Hsuan Ho, Lee-Feng Chien, and Lin-Shan Lee. 1998. Statistics-based segment pattern lexicon: A new direction for Chinese language modeling. In *Proceedings of the IEEE International Conference on Acoustic, Speech, Signal Processing*. 169–172.

Lei Zhang, Zhou Ming, Changning Huang, and Mingyu Lu. 2000. Approach in automatic detection and correction of errors in Chinese text based on feature and learning. In *Proceedings of the 3rd World Congress on Intelligent Control and Automation*. 2744–2748.

Received August 2014; revised January 2015, June 2015; accepted June 2015