# Correction of Erroneous Characters in Chinese Sentence Analysis

***Andi Wu***
andiwu@microsoft.com
***George Heidorn***
georgehe@microsoft.com
***Zixin Jiang***
jiangz@microsoft.com
***Terence Peng***
tpeng@microsoft.com
Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399
U.S.A
(425) 936-0985

## Abstract

This paper presents a method of automatically detecting and correcting erroneous characters in electronic Chinese texts. The characters to be corrected are those that are easily confused with other characters because of their phonological or graphical resemblance. The correction takes place as an integral part of syntactic analysis, where both the original character in the text and some replacement character(s) are considered, and the character that ends up in the final parse is identified as the correct one. The accuracy of this method substantially surpasses existing proofreaders in both recall and precision in the category of substitution errors. A demo of the system is available.

本文介绍一种汉语文本自动别字纠错系统。该系统所纠正的字均为因音同或形似而易混淆的字。纠错在句子分析过程中进行：句子分析器会同时考虑原文中的字及该字混淆集中的字，正确的字为最佳分析结果之结构树中所出现的字。使用该方法所得到的纠错召回率和纠错精确率均大大超出现有的中文校对系统。

## Introduction

Written Chinese uses more than 5,000 distinctive characters and many of them have similar sounds or shapes. As a result, users of Chinese tend to mistake one character for another when typing if these two characters are similar phonetically or graphically. When speech input is used, the confusion transfers from the human user to the computer: the speech recognizer may make mistakes because of the phonetic similarities between

characters. The automatic detection and correction of those erroneous characters are crucial to the usability of any word processing software for Chinese. They are also required for robust parsing where sentences containing wrong characters can also be successfully analyzed.

Existing Chinese word processors, such as Microsoft Word 2000 and IBM WordPro '97, already have proofreaders that can detect and occasionally correct wrong characters. However, their recall and precision rates are fairly low (both below 65%). These proofreaders are usually based on a word segmenter, an n-gram model, and some local rules and heuristics. Their performance is hard to improve beyond a certain point because they do not analyze the structure of a sentence and therefore have no access to the larger context in which a character is used.

The approach to be discussed in this paper treats typo correction as an integral part of syntactic analysis. It is motivated by the reality that we must be able to process sentences that have not been edited and therefore may contain incorrect characters. Implemented in NLPWin, the general-purpose language understanding system being developed at Microsoft Research (Heidorn 2000, Jensen et al 1993), it assumes the existence of a parser, a Chinese grammar, and the word segmentation mechanism described in Wu and Jiang (1998). In addition, we need a list of confusable pairs.

A confusable pair consists of two characters that are often mistaken for each other. Examples of such pairs are 那/哪, 帐/账, 清/青, 杨/扬, 于/与, etc. The actual system we have implemented can also handle confusable triples, quadruples and so on, but we will focus on confusable pairs in this paper for the ease of exposition. The actual members of the confusable pairs list can be customized for different purposes. If the text to be proofread is created with a phonologically-based input method, the list is expected to contain pairs that have identical sounds. Texts created with stroke-based input method, on the other hand, will require pairs that are similar in shape. When used in a pedagogical setting, schoolteachers can construct a list that contains only those pairs that they are trying to make their students distinguish. When integrated with a statistically-based input method, the list can also be generated dynamically from the candidates suggested by the statistical model. In this case, the confusable pair could be the top *n* candidates that have competing scores.

## The correction process

This section covers the computational details of the correction process. We will start with a general description of the approach and then give a step-by-step illustration of the mechanism by going through an example.

### General description

As we have mentioned in the introduction, the original motivation for this project is to have a robust parser that can successfully analyze a sentence even if the sentence contains

erroneous characters. Since the analysis of a sentence involves both lexical processing and syntactic processing, it is quite natural that some errors will be corrected during lexical analysis, some during syntactic analysis, and some during both. The actual process depends on the nature of the erroneous character.

In cases where the intended character is part of a multiple-character word, the wrong character in its place will make it impossible for this word to be looked up in the dictionary. In the following sentence, for example, the wrong character 彷 in the place of 仿 will prevent us from looking up the two-character word 仿佛.

人们没有理她，**彷**佛她并不存在。

In this case, the error can be easily corrected if we have the confusable pair 彷/仿 in our list. Knowing that 彷 can be mistakenly used in place of 仿, we will consider both possibilities in our dictionary lookup. Given the character 彷, we find that it cannot be used as a word by itself and it is not able to combine with any of the neighboring characters to form a word. In other words, the span of the sentence would be broken at this character. The character 仿, on the other hand, can combine with 佛 to form the word 仿佛. Since only the use of 仿 can result in a spanning lexical array, we have sufficient evidence that 彷 should be replaced by 仿 in this sentence. Consequently, 彷 will be removed from the lexical array even before syntactic analysis begins.

Sometimes, the erroneous character is an independent word but it is meant to be part of a multiple-character word. In the following sentence, 辨 is mistakenly used for 辩, which is part of the word 辩论. However, both 辨 and 论 can be independent words themselves and therefore the wrong character will not result in a non-spanning lexical array.

目前律师和法院正在就这个棘手案件进行**辨**论 。

If 辨/辩 is a confusable pair in our list, we will try to look up both 辨论 and 辩论 in the dictionary. We will find that only 辩论 is a word in the dictionary. In addition, it has an "IgnoreParts" attribute, which means that, once these two characters are combined into a single word, it will no longer be possible for the component characters of this word to be independent words. In our system, all the records that are subsumed by an "IgnoreParts" word are removed from the lexical array unless this word overlaps with another word. The wrong character 辨 in this sentence is not a component of 辩论 but it is in the span of 辩论 and therefore structurally subsumed by it. As a result, it will be removed from the lexical array before syntactic analysis, and only the correct character will end up in the final parse. The use of "IgnoreParts" in this way is not completely safe, since theoretically 辨 and 论 can be two independent words in the sentence. But so far there has not been any empirical evidence that this is a problem.

In cases where the multiple-character word does not have the "IgnoreParts" attribute, we will keep both this word and its components while lowering the probabilities of the

components. Consider the following example, assuming that 定/订 is in the list of confusable pairs.

我冲到楼下的咖啡厅，猛喝了几杯咖啡，然后直奔旅行社**定**票。

Here the typist mistook 定 for 订. Looking up both 定票 and 订票 in the dictionary, we find that 订票 is a word while 定票 is not. However, 订票 does not have the "IgnoreParts" attribute and both 定 and 票 are independent words. We cannot remove the words subsumed by 订票 in this case. Instead, we will raise the probability of 订票 and lower the probabilities of 定, 订, and 票. During syntactic analysis, we may have several spanning parses, each consisting of different words. However, the parse containing 订票 will have the highest score and be selected as the best parse. As a result, only 订 will be kept in the final analysis.

There are also cases where both the wrong character and the intended character can combine with their neighbors to form words. In the following example, 注 is used in place of 主, but both 注 and 主 can combine with 意 to form words.

我遇到极大的困难，你给我出个**注**意吧。

Assuming that 注/主 is one of our confusable pairs, we will look up both 注意 and 主意. There is no lexical evidence here to tell us whether 注 or 主 is the correct character in this sentence. However, the choice is very clear at the syntactic level. 注意 is a verb and 主意 is a noun. Since the grammar requires a noun in the slot occupied by 注意/主意, only the use of 主意 in this sentence can produce a spanning parse. If the alternative words have the same parts of speech and share all other lexical properties, then the choice will be random unless we have semantic information to prefer one to the other.

Another case where only syntactic information can help is the one where both the wrong character and the intended character are independent words and neither of them can combine with neighboring words to form a longer word. The following sentence is such an example.

那里有二十余幢中外投资大厦相继**与**年底前开工。

In this sentence, the wrong character 与 is used instead of 于. With 与/于 as one of the confusable pairs in our system, we will put both words in the chart and see which of them can "survive" the parsing process. It turns out that we can get a spanning parse only if 于 is used. We can then conclude that the intended character is 于 rather than 与.

**Step-by-step illustration**

Now we will go through the actual computational steps, using the following input sentence as an example.

他**再那**里**泡**制了这篇混**帐**文章？

This sentence contains 4 wrong characters: 再 for 在, 那 for 哪, 泡 for 炮, and 帐 for 账. Now assume that these 4 pairs are in the list of confusables in our system. The correction procedure will go as follows in the process of sentence analysis.

(1) Segment the input sentence into single characters and look up each of them from the dictionary. For our example sentence, this results in the following array of lexical records. It is displayed as a word lattice because there can be more than one record in a given span of the sentence.

```
他 PRON1
:: 再 ADV1
:::: 那 PRON2
:::: 那 CONJ1
:::::: 里 NOUN1
:::::::: 泡 NOUN2
:::::::: 泡 VERB1
:::::::::: 制 NOUN3
:::::::::: 制 VERB2
:::::::::::: 了 VERB3
:::::::::::: 了 FUNCW1
:::::::::::::: 这 PRON3
:::::::::::::::: 篇 NOUN4
:::::::::::::::::: 混 VERB4
:::::::::::::::::::: 帐(n)
:::::::::::::::::::::: 文 NOUN5
:::::::::::::::::::::::: 章 NOUN6
```

Notice that 帐 is not an independent word in the dictionary. So instead of having a regular part of speech, it is marked with (n), which means that its "potential" part of speech is a noun.

(2) For each character, check the list of confusables to see if it is a member of one of the pairs. If so, retrieve the other member of the pair, look up this character from the dictionary, and add the new lexical record(s) to the word lattice. Below is the expanded word lattice where we see the additional records for 哪, 在, 炮 and 帐 (underlined).

```
 他 PRON1
:: 再 ADV1
:: 在 VERB1
:: 在 ADV2
:: 在 PREP1
:::: 那 PRON2
:::: 那 CONJ1
:::: 哪 PRON3
:::::: 里 NOUN1
:::::::: 泡 NOUN2
:::::::: 泡 VERB2
:::::::: 炮 NOUN3
:::::::::: 制 NOUN4
:::::::::: 制 VERB3
:::::::::::: 了 VERB4
:::::::::::: 了 FUNCW1
:::::::::::::: 这 PRON4
:::::::::::::::: 篇 NOUN5
:::::::::::::::::: 混 VERB5
:::::::::::::::::::: 帐(n)
:::::::::::::::::::: 账 NOUN6
:::::::::::::::::::::: 文 NOUN7
:::::::::::::::::::::::: 章 NOUN8
```

(3) Look up multiple-character words from the dictionary, using both the original characters and the characters added in Step 2. Add the new records (underlined) to the word lattice.

```
他 PRON1
:: 再 ADV1
:: 在 VERB1
:: 在 ADV2
:: 在 PREP1
:::: 那里 PRON5
:::: 哪里 PRON6
:::: 那 PRON2
:::: 那 CONJ1
:::: 哪 PRON3
:::::: 里 NOUN1
:::::::: 炮制 VERB6
:::::::: 泡 NOUN2
:::::::: 泡 VERB2
:::::::: 炮 NOUN3
:::::::::: 制 NOUN4
:::::::::: 制 VERB3
:::::::::::: 了 VERB4
:::::::::::: 了 FUNCW1
:::::::::::::: 这 PRON4
:::::::::::::::: 篇 NOUN5
:::::::::::::::::: 混账 NOUN9
:::::::::::::::::: 混 VERB5
:::::::::::::::::::: 帐(n)
:::::::::::::::::::: 账 NOUN6
:::::::::::::::::::::: 文章 NOUN10
:::::::::::::::::::::: 文 NOUN7
:::::::::::::::::::::::: 章 NOUN8
```

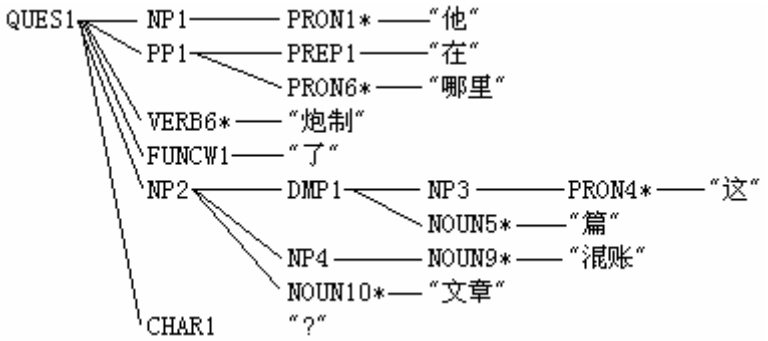
We see that neither member of the 再/在 pair is able to combine with its neighboring characters to form words. This is a case where we have two competing single-character words. On the other hand, both members of the 那/哪 pair are able to combine with the next character to form 那里 and 哪里 respectively. This is a case where we have two competing multiple-character words. Of the other two pairs, 泡/炮 and 帐/账, only one member in each pair can form a multiple-character word with its neighbor, resulting in 炮制 and 混账.

(4) For each multiple-character word in the word lattice, check to see if it has the "IgnoreParts" attribute. If so, the single characters subsumed by this word will be removed. Otherwise, the subsumed single characters will be assigned low probabilities. Here is the word lattice after the removal and probability lowering: (Low probability words are grayed out.)

```
他 PRON1
:: 再 ADV1
:: 在 VERB1
:: 在 ADV2
:: 在 PREP1
:::: 那里 PRON5
:::: 哪里 PRON6
:::::::: 炮制 VERB6
:::::::: 泡 NOUN2
:::::::: 泡 VERB2
:::::::: 炮 NOUN3
:::::::::: 制 NOUN4
:::::::::: 制 VERB3
:::::::::::: 了 VERB4
:::::::::::: 了 FUNCW1
:::::::::::::: 这 PRON4
:::::::::::::::: 篇 NOUN5
:::::::::::::::::: 混账 NOUN9
:::::::::::::::::::::: 文章 NOUN10
```

By this time, it has already become quite clear that 帐 is a wrong character in the original text and it should be replaced by 账, because only the latter was able to survive lexical processing. 泡 is most likely to be wrong, too, since it has low probability while its competitor 炮 is part of a two-character word that has high probability. The remaining decisions are to be made in syntactic processing.

(5) Submit the lexical array to syntactic processing where every lexical record in the word lattice will be in the parser's chart and compete with each other until a successful parse is found:




We can see that the four replacement characters, 在, 哪, 炮 and 账, are in the tree while all the wrong characters have disappeared.

(6) Output the leaves of the tree as the corrected sentence:

他在哪里炮制了这篇混账文章？

## Evaluation and Discussion

### Evaluation

The typo correction mechanism described above has been tested on more than 50 frequent pairs of confusable characters using randomly extracted sentences. For each pair A/B, we first assume that the correction is in the direction of A B and then in the direction of B A. In each case, we call the character on the left side the "text character" and the one on the right side "replacement character". Following standard practice, we ran both a recall test and a precision test on each pair in both directions.

The extracted sentences are hand-checked to make sure that they do not contain erroneous characters. In the recall test, we did the following for each A B where A is the text character and B is the replacement character.

(1) Collect 100 sentences that contain B;
(2) Replace all the instances of B with A;
(3) Parse the modified sentences with the correction mechanism and output the corrected sentences.
(4) Compare the output with the original sentences to see how many errors have been corrected.
(5) Recall rate = number of sentences corrected / total number of sentences.

In the precision test, we did the following for each A B:

(1) Collect 100 sentences that contain A;
(2) Parse those sentences with the correction mechanism.

(3) Compare the output with the original sentences to see how many sentences remain unchanged (where the As have not been mistakenly changed to Bs).
(4) Precision rate = number of sentences unchanged / total number of sentences.

Here is a representative sample of the test results:

| Text Character | Replacement Character | Recall Rate | Precision Rate |
|---|---|---|---|
| 那 | 哪 | 71% | 90% |
| 与 | 于 | 87% | 83% |
| 帐 | 账 | 98% | 90% |
| 嘛 | 吗 | 82% | 96% |
| 辨 | 辩 | 95% | 74% |
| 煅 | 锻 | 100% | 100% |
| 彷 | 仿 | 100% | 100% |
| 作 | 做 | 30% | 100% |
| 在 | 再 | 70% | 97% |
| 坐 | 座 | 86% | 93% |

The overall precision rate across all the pairs tested is 86.9% and the overall precision rate is 96.3.

**Discussion**

Some interesting observations can be made on the results in the above table.

(1) 煅　锻 and 彷　仿 have perfect scores on both recall and precision. This is due to the fact that all the 4 characters involved here are usually used as parts of multiple-character words rather than words by themselves. In a given context, only one member of the pair will be able to combine with neighboring characters to form words. In these cases, the correction can be basically done at the lexical level and the result is seldom affected by syntactic analysis.
(2) 那　哪 has a fairly high precision score but a low recall score. This is because 那 can be used wherever 哪 can be used unless there is clear evidence, such as a question mark, that indicates that the sentence is a wh-question. (This is comparable to there/where in English.) A similar situation is found in 嘛　吗 where 吗 is only licensed in a yes/no question.
(3) 辨　辩 has a high recall score but a low precision score. Looking at the data, we find that, in all the cases where 辨 is mistakenly changed to 辩, 辨 is used as an independent word. Since 辩 can also be used as an independent word with the same part of speech and other grammatical attributes, the chances of their appearing in the final parse are almost equal, making the correction close to random.

(4) 在　再 has a low recall score but a high precision score. Both 在 and 再 are very active independent words and the grammatical contexts in which 再 appears are a proper subset of the contexts where 在 appears. Therefore it is not so easy to switch from 在 to 再.
(5) 作　做 has an extremely low recall score. This is because 作 is becoming a free variation of 做 in contemporary written Chinese. In many words and grammatical contexts where 做 should be used, 作 can also be used. In other words, the confusion is being accepted by the general public. Therefore, this pair should probably be removed from the list of confusables for practical purposes.

The test results also reveal the following limitations of the current approach:

(1) It assumes that every sentence can be successfully parsed as long as the wrong characters have been corrected. Therefore, we will have a problem if the input is not grammatical or is beyond the grammar coverage of the parser. In this case, the correction is guaranteed to work only if the choice is already clear during lexical processing.
(2) Some sentences can produce spanning parses in spite of the fact they contain erroneous characters. This usually happens when the wrong character is an independent word and it happens to have the right syntactic properties to fit into a parse.
(3) In cases where both the text character and the replacement character can combine with their neighbors to form multiple-character words, the success of the correction is more by chance if the words they form are of the same length and with the same part of speech and other grammatical properties.
(4) The current approach only covers the correction of "substitution errors" where the wrong character(s) and intended character(s) span the same position(s) in a sentence. It does not cover cases where characters are accidentally added or omitted.
(5) It still remains to be seen how long the list of confusable pairs can become. Since every confusable character in a sentence introduces some additional records to the lexical array, a sentence containing too many confusable characters will have a more complex word lattice, which results in more ambiguity and noise and makes parsing more difficult.

Despite those limitations, however, the current approach has been able to cover cases where no solution was available before and achieved accuracy rates that are not matched by any existing proofreaders. While it does not solve every problem in Chinese proofreading, it does provide a better framework in which more problems can have a solution.

## References

Heidorn, G. E. (2000): Intelligent Writing Assistance. In Dale R., Moisl H., and Somers H. (eds*.), A Handbook of Natural Language Processing: Techniques and Applications for*

*the Processing of Language as Text.* Marcel Dekker, New York, 1998 (published in August 2000), pages 181-207.

Guo, Zhili and Zhaoming Qiu (1997) The candidate suggestion algorithm in a practical Chinese error check system. In Chen, Liwei et al (Eds) *Language Engineering*, pages 325-331, Tsinghua University, Beijing.

Jensen, K., Heidorn G., and Richardson S. (eds.) (1993): *Natural Language Processing: The PLNLP Approach*, Boston, Kluwer.

Song, Rou, L. Ouyang and C.Qiu (1998) The functions of a Chinese proofreading system. In Publication and Printing, Vol. 3, 1998.

Sun, Cai and Zhengsheng Luo (1997). Reseach on the lexical errors in Chinese Text. In Chen, Liwei et al (Eds) *Language Engineering*, pages 319-324, Tsinghua University, Beijing.

Wu, Andi and Zixin Jiang (1998). Word segmentation in sentence analysis. In *Proceedings of the 1998 International Conference on Chinese Information Processing*, pages 169-180, Beijing, China.

Zhang, Yangsen and Bingqing Ding (1997) Research and practice on the lexical error detecting system based on "bundling and filtering" in Chinese Text Automatic Proofread. In *Proceedings of the 1998 International Conference on Chinese Information Processing*, pages 392-397, Beijing, China.