Research Article

Effectiveness of Context-Aware Character Input Method for Mobile Phone Based on Artificial Neural Network

Masafumi Matsuhara and Satoshi Suzuki

1 Department of Software and Information Science, Iwate Prefectural University, 152-52, Takizawa, Iwate 020-0193, Japan
2 Supernet Department, System Consultant Co., Ltd., 2-14-6, Kinshi, Sumida, Tokyo 130-0013, Japan

Correspondence should be addressed to Masafumi Matsuhara, masafumi@iwate-pu.ac.jp

Received 10 February 2012; Revised 19 April 2012; Accepted 26 April 2012

Academic Editor: Cheng-Hsiung Hsieh

Opportunities and needs are increasing to input Japanese sentences on mobile phones since performance of mobile phones is improving. Applications like E-mail, Web search, and so on are widely used on mobile phones now. We need to input Japanese sentences using only 12 keys on mobile phones. We have proposed a method to input Japanese sentences on mobile phones quickly and easily. We call this method number-Kanji translation method. The number string inputted by a user is translated into Kanji-Kana mixed sentence in our proposed method. Number string to Kana string is a one-to-many mapping. Therefore, it is difficult to translate a number string into the correct sentence intended by the user. The proposed context-aware mapping method is able to disambiguate a number string by artificial neural network (ANN). The system is able to translate number segments into the intended words because the system becomes aware of the correspondence of number segments with Japanese words through learning by ANN. The system does not need a dictionary. We also show the effectiveness of our proposed method for practical use by the result of the evaluation experiment in Twitter data.

1. Introduction

Ordinary Japanese sentences are expressed by two kinds of characters, that is, Kana and Kanji. Kana is Japanese phonogramic characters and has about fifty kinds. Kanji is ideographic Chinese characters and has about several thousand kinds. Therefore, we need to use some Kanji input methods in order to input Japanese sentences into computers. A typical method is the Kana-Kanji translation method. This method translates nonsegmented Kana sentences into Kanji-Kana mixed sentences. Since one Kana character is generally inputted by combination of a few alphabets, this method needs twenty six keys for the alphabets.

Recently, performance of mobile computing devices is greatly improving. We consider that the devices are grouped into two by their quality. One gives importance to easy operation, the other gives importance to good mobility. Mobile phones are usable as mobile computers and belong to the latter group. Their mobility is very good because typical size of them is small. However, a general mobile phone has only 12 keys, which are 0, 1, . . . , 9, *, and #, because of the limited size. A growing number of Smartphones, for example, iPhones, Blackberries, and so on, have full QWERTY keyboards. It is not easy to press the intended key because the key size is small. Moreover, a user needs to press a few keys per Kana character since one Kana character generally consists of a few alphabets. Therefore, we focus on 12 keys layout on the mobile phones.

The letter cycling input method is most commonly used for the input of sentences on mobile phones. In this input method, a chosen key represents a consonant, and the number of pressing it represents a vowel in Japanese. For example, the chosen key “7” represents “m”, and three presses of the key represent “u”. Then, the number of key presses is three for the input character “む (mu)”. Since this input method needs several key presses per Kana character, it is troublesome for a user. Opportunities and needs are rapidly increasing to input Japanese sentences into a small device such as a mobile phone since performance of mobile phones...
increases to select the intended word because there are many word candidates. Therefore, we focus on a number-Kanji translation method without prediction.

We have proposed a number-Kanji translation method based on artificial neural network (ANN) [9]. The system becomes aware of the correspondence of number segments with Japanese words through learning by ANN. Then, the system translates an inputted number string by ANN. The system does not use dictionaries for translation. Therefore, the system may translate the number-segments into unknown words without dictionaries. Moreover, the system requires the only fixed memory determined by the size of ANN. Because of reduced memory requirement, we consider that our proposed method is especially suitable for a mobile phone.

This paper shows the outline of the number-Kanji translation, the processes of our proposed method, the evaluation experiment, its result, and the effectiveness of our proposed method for practical use.

2. Outline of Number-Kanji Translation

Figure 3 shows an example of the number-Kanji translation. A user inputs the number-string “41210213139” for the Kanji-Kana mixed sentence “大会を開催する (The meeting is held).”. A user is able to input rapidly and easily because one key stroke corresponds to one Kana character. Figure 3 shows a number-Kanji mixed sentence “大会を開催する (The meeting is held).” A user is able to input rapidly and easily because one key stroke corresponds to one Kana character.

A user inputs a string of numbers corresponding to the pronunciation of an intended Japanese sentence based on Figure 1. The Kana-Kanji translation method translates a Kana sentence, whereas the number-Kanji translation method translates a string of numbers. A key pressed on the keypad of 12 keys represents a line of the 50-sound table of Kana, which is the Japanese syllabary. Figure 2 shows the 50-sound table. It is set in a five-by-ten matrix. The matrix has five vowels and ten consonants. Almost all Kana characters are composed of a consonant plus a vowel. A user is able to input one Kana character per key press.

Figure 1 shows the correspondence of the number with Kana characters: for example, the key “4” represents “た (ta)” or “ち (ti)” or “つ (tu)” or “つ (te)” or “と (to)” of Kana characters. The characters in parentheses represent the pronunciation of Kana. Then, a number character of 12 keys generally corresponds to a consonant. Since the vowel information degenerates, the string of numbers has ambiguity: for example, the number-string “4121” corresponds to not...
only the Kana characters "たいかい (taikai)" but also "ていこう (teikou)", "とうこう (toukou)", and so on. Moreover, a string of Kana character means some Japanese words: for example, the Kana characters "たいかい (taikai)" mean not only the Japanese word "大会 (the meeting)" but also "退会 (withdrawal)", "大海 (ocean)", and so on. Our proposed method uses ANN for the disambiguation.

The user presses the key "∗" for a voiced consonant and a p-sound in our proposed method. For example, the user inputs the number-string "4∗12" for the Japanese word "大工 (a carpenter)" of which the pronunciation is "だいiku (ta∗iku)" in Japanese. However, "da" is translated into "4∗" and the "4∗" also corresponds to "ta∗" in the system. Therefore, "daiku" is expressed as "ta∗iku" in this paper).

3. Processes

Our proposed method has the learning stage and the translation stage. Figure 4 shows the procedure in the translation stage. The procedure consists of the division process, the translation process, and the combination process in this order.

3.1. Division Process. Our proposed method uses ANN, and the size of ANN needs to be fixed basically. A user inputs a string of numbers corresponding to the pronunciation of an intended Japanese sentence. It is difficult to design ANN because the length of a natural language sentence is indefinite and a Japanese sentence is not segmented. Therefore, the system based on our proposed method divides the inputted number-string into the number-segments with a fixed length.

Figure 5 shows an example of the division process. The inputted number-string is divided into 11 segments, that is, from segment 1 to segment 11. The fixed length of each segment is 4 in Figure 5.

It is easy to design ANN because the length of the segments is fixed. However, the segmentations are not always correct. The segments may include incorrect words.

Therefore, the system needs to select the correct words and to combine them for making up the Japanese sentence intended by the user in the combination process.

3.2. Translation Process. The system becomes aware of the correspondence of number-segments with Japanese words through learning by ANN in the learning process. The system translates each divided segment by the ANN. The system needs to translate the correct segments into the correct Japanese words and to decide the incorrect segments.

Figure 6 shows an example of the translation process. Each segment divided in the division process is translated by ANN. The segment 1 needs to be translated into the correct word "大会 (the meeting)" because its segmentation is correct. The segment 2 needs to be decided as the incorrect segment because its segmentation is incorrect. Then, the segment 2 is translated into "FFFF" as a noncharacter code in Figure 6.

3.3. Combination Process. The system based on our proposed method makes up the Japanese sentence to combine the translation result because the translation result is divided into segments.

Figure 7 shows an example of the combination process. The segment 2, the segment 11, and soon are decided as the incorrect words. Then, the system makes up the Japanese...
sentence “大会を開催する” to combine the segment 1, the segment 5, the segment 6, and the segment 10 in Figure 7.

3.4. Learning Stage. The learning stage is performed independent of the translation stage. The system becomes aware of the correspondence of number-segments with Japanese words through learning by ANN.

We use multilayer feed-forward neural network trained by error backpropagation. The excitations propagate in a single direction, from the input layer to the output layer, through multiple intermediate layers, often called hidden layers. The connection weights, which mimic the synapses, are initialized with random values and gradually trained for the task in hand using a gradient descent training algorithm. The most common one is known as error backpropagation [10]. Thus, the functionality of the network is stored among the connection weights of different neuron nodes in a distributed manner.

The structure of ANN is shown in Figure 8. A number-string is inputted to the input layer as the input value. The number-string has 12 kinds of characters, that is, 0, 1, ..., 9, *, and #. Since each input value is a binary digit, the input layer needs 4 nodes per character. The number-string consists of the forward number-string and the number-segment. A forward number-string has $l$ characters. A number-segment has $m$ characters. Therefore, the input layer has $4 \times (l + m)$ nodes. A Japanese word is outputted to the output layer as the output value. The output value is a binary digit also. Since a Japanese character needs 2 Bytes = 16 nodes, the output layer has $16 \times n$ nodes for $n$ Japanese characters. The network is adjusted by evaluating the difference of a predicted character and a given character as nodes (=binary digits) in the output layer.

For example, the correspondence of the number-segment “4121” with the Japanese word “大会” is learned by ANN. Then, the system is able to translate the number-segment “4121” into the Japanese word “大会” without a dictionary. Not only a segment but also its forward number-string is learned by ANN. For example, the forward number-string “2131” of the segment “39” is learned. Then, the backward segment “39” of the number-string “2131” is able to translate into the correct word “する”. Thus, our proposed method uses a context.

4. Evaluation Experiment

The system based on our proposed method has been developed for an experiment. The system is not able to make up the correct Japanese sentence in the combination process.
First of all, we evaluated the translation accuracy in the translation process. Therefore, we considered that 6% translation error for non-character code is not always large. The translation accuracy of Japanese Kanji translation method is about 95% per character in general. Therefore, we consider that 6% translation error for Japanese character code is not always large. The Kanji-Kanji translation method translates a Kanji sentence, whereas our proposed method translates a string of numbers. It is difficult to translate a number-string because a number-string is more ambiguous than a Kanji sentence. The accuracy of the number-Kanji translation method is about 85% per character in our previous work [3]. Therefore, the accuracy of our proposed method is never low even though the accuracy is per node. We consider that the accuracy achieves a practical level.

Table 1: Experiment data.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of characters</td>
<td>55,951</td>
</tr>
<tr>
<td>No. of different words</td>
<td>4,199</td>
</tr>
<tr>
<td>No. of character code segments</td>
<td>20,674</td>
</tr>
<tr>
<td>No. of noncharacter code segments</td>
<td>28,565</td>
</tr>
</tbody>
</table>

Table 2: Parameter of ANN.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of input nodes</td>
<td>40</td>
</tr>
<tr>
<td>No. of hidden nodes</td>
<td>144</td>
</tr>
<tr>
<td>No. of output nodes</td>
<td>144</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.01</td>
</tr>
</tbody>
</table>

Table 3: Accuracy of translation per node.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Japanese character code</td>
<td>93.4 [%]</td>
</tr>
<tr>
<td>Noncharacter code</td>
<td>98.8 [%]</td>
</tr>
<tr>
<td>Total</td>
<td>96.5 [%]</td>
</tr>
</tbody>
</table>

Table 4: Mean number of erroneous node per segment.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Japanese character code</td>
<td>10.64</td>
</tr>
<tr>
<td>Noncharacter code</td>
<td>1.97</td>
</tr>
<tr>
<td>Total</td>
<td>5.62</td>
</tr>
</tbody>
</table>

The data is divided into 5 sets for K-fold cross-validation. Each of the 4 sets is used to train the network, and the rest 1 set is used to test.

4.1. Experiment Data and Procedure. The data for the experiment is text a user inputted on Twitter (an online social networking service http://twitter.com/). The detail is shown in Table 1. The character code segments correspond to the correct words. They have to be translated into the Japanese words. The noncharacter code segments correspond to the incorrect words. They have to be translated into “FFFF” in the translation process.

The parameter of ANN is shown in Table 2. The input nodes are for the divided number-segments and the forward number-string. The max length of the segments is 6 (=m in Figure 8), and the length of the forward string is 4 (=l in Figure 8). The value is decided by the preliminary experiments. The number of input nodes is 40 because a number character needs 4 nodes in the network. The output nodes are for the character codes of the Japanese words. The max length of the words is 9 (=n in Figure 8), and a Japanese character needs 16 nodes (2 Bytes) in the network. Then, the number of output nodes is 144. The number of hidden nodes is equal to the number of output nodes. The learning rate is 0.01.

The data is divided into 5 sets for K-fold cross-validation. Each of the 4 sets is used to train the network, and the rest 1 set is used to test.

4.2. Results and Considerations. First of all, we evaluated the root mean square errors (RMSEs) in the learning stage for confirmation of the learning times. Figure 9 shows RMSE for each set of 5 sets for K-fold cross-validation in the learning stage. In Figure 9, the errors are decreasing as the learning times are increasing. The value of RMSE is below 0.005, and the changes are convergent finally. Therefore, it is shown that the system is able to learn the data normally. 10,000 epochs are sufficient for the training of the data.

Table 3 shows the mean rate for the correct translation per node in the network of the Japanese character code, the noncharacter code, and total in the translation process. In Table 3, the accuracy of translation for noncharacter code is higher than that for Japanese character code. This is because the segments of noncharacter code are larger than ones of Japanese character code. Ordinarily, the translation accuracy tends to be higher when the data is large for the learning.

The translation accuracy of Japanese Kanji translation method is about 95% per character in general. Therefore, we consider that 6% translation error for Japanese character code is not always large. The Kanji-Kanji translation method translates a Kanji sentence, whereas our proposed method translates a string of numbers. It is difficult to translate a number-string because a number-string is more ambiguous than a Kanji sentence. The accuracy of the number-Kanji translation method is about 85% per character in our previous work [3]. Therefore, the accuracy of our proposed method is never low even though the accuracy is per node. We consider that the accuracy achieves a practical level.

Table 4 shows the mean number of the erroneous nodes per segment for the Japanese character code, the noncharacter code, and total. The non-character code means the segmentation is wrong, and the number-segment does not correspond to a Japanese word. The system needs to distinguish the segments with Japanese character code from ones with non-character code. The distinction is never easy because the non-character code segment may correspond to another Japanese word.

In Table 3, the accuracy of translation for non-character code is 98.8%. In Table 4, the mean number of erroneous nodes is 1.97. Then, the translation accuracy of the segment for non-character code is high. The accuracy for Japanese character code is 93.4%. Although the rate is high, the translation result has errors. The mean number of erroneous nodes is 10.64 in Table 4. The value is low relatively because
the size of the output nodes is 144. Therefore, we consider that it is possible to translate the erroneous nodes into the correct words by increasing learning data or adding the correction process and so on.

We are able to calculate the total number of links in the network. The number of links is defined as

\[
\text{no. of links} = (\text{no. of input nodes} + 1) \\
\times \text{no. of hidden nodes} \\
+ (\text{no. of hidden nodes} + 1) \\
\times \text{no. of output nodes},
\]

where “+1” means an additional node for a bias of ANN. The total number of links in the system of the evaluation experiment is calculated as

\[
(40 + 1) \times 144 + (144 + 1) \times 144 = 26,784. \quad (2)
\]

If the size for a weight is 4 Bytes per link in the network, the size of memory is about 107 KB. The size is small and fixed. The memory size does not change when the learning data increases. Therefore, it is easy to implement our proposed method on a mobile phone.

5. Conclusion

In this paper, we proposed a context-aware number-Kanji translation method using ANN and have shown the effectiveness of the method by the actual experiment for practical use.

The algorithm enables to input one Kana character per key stroke. Then, a user is able to input a Japanese text rapidly and easily. However, a string of numbers inputted by the user is ambiguous. Our proposed method disambiguates the number-string and translates it into the Japanese sentence intended by the user using ANN. The system becomes aware of the correspondence of number-segments with Japanese words through learning. Therefore, the system is able to translate the number-string into the intended sentence by ANN without a dictionary. The system requires the fixed memory determined by the size of ANN. Because of reduced memory requirement, our proposed method is especially suitable for a mobile phone.

In the experiment, we use Twitter data to confirm the effectiveness of our proposed method for practical use. The accuracy of the translation per node is high. The mean number of the erroneous nodes is about 11 per segment for Japanese character code. The value is low in comparison with the size of the output nodes in the network. Therefore, we consider that it is possible to translate the erroneous segments into the correct words. By the actual experiment, it is shown that our proposed method is effective for practical use.

One of future works is to add the correction process for recovering the erroneous nodes. Then, we need to evaluate the translation accuracy in the combination process and compare with current popular methods.

References


Submit your manuscripts at
http://www.hindawi.com