Combining Trigram and Winnow in Thai OCR Error Correction

Surapant Meknavin
National Electronics and Computer Technology Center
73/1 Rama VI Road, Rajthevi, Bangkok, Thailand
surapan@nectec.or.th

Boonserm Kijsirikul, Ananlada Chotimongkol and **Cholwich Nuttee**
Department of Computer Engineering
Chulalongkorn University, Thailand
fengbks@chulkn.chula.ac.th

Abstract

For languages that have no explicit word boundary such as Thai, Chinese and Japanese, correcting words in text is harder than in English because of additional ambiguities in locating error words. The traditional method handles this by hypothesizing that every substrings in the input sentence could be error words and trying to correct all of them. In this paper, we propose the idea of reducing the scope of spelling correction by focusing only on dubious areas in the input sentence. Boundaries of these dubious areas could be obtained approximately by applying word segmentation algorithm and finding word sequences with low probability. To generate the candidate correction words, we used a modified edit distance which reflects the characteristic of Thai OCR errors. Finally, a part-of-speech trigram model and Winnow algorithm are combined to determine the most probable correction.

1 Introduction

Optical character recognition (OCR) is useful in a wide range of applications, such as office automation and information retrieval system. However, OCR in Thailand is still not widely used, partly because existing Thai OCRs are not quite satisfactory in terms of accuracy. Recently, several research projects have focused on spelling correction for many types of errors including those from OCR (Kukich, 1992). Nevertheless, the strategy is slightly different from language to language, since the characteristic of each language is different.

Two characteristics of Thai which make the task of error correction different from those of other languages are: (1) there is no explicit word boundary, and (2) characters are written in three levels; i.e., the middle, the upper and the lower levels. In order to solve the problem of OCR error correction, the first task is usually to detect error strings in the input sentence. For languages that have explicit word boundary such as English in which each word is separated from the others by white spaces, this task is comparatively simple. If the tokenized string is not found in the dictionary, it could be an error string or an unknown word. However, for the languages that have no explicit word boundary such as Chinese, Japanese and Thai, this task is much more complicated. Even without errors from OCR, it is difficult to determine word boundary in these languages. The situation gets worse when noises are introduced in the text. The existing approach for correcting the spelling error in the languages that have no word boundary assumes that all substrings in input sentence are error strings, and then tries to correct them (Nagata, 1996). This is computationally expensive since a large portion of the input sentence is correct. The other characteristic of Thai writing system is that we have many levels for placing Thai characters and several characters can occupy more than one level. These characters are easily connected to other characters in the upper or lower level. These connected characters cause difficulties in the process of character segmentation which then cause errors in Thai OCR.

Other than the above problems specific to Thai, real-word error is another source of errors that is difficult to correct. Several previous works on spelling correction demonstrated that

Figure 1: No explicit word delimiter in Thai

feature-based approaches are very effective for solving this problem.

In this paper, a hybrid method for Thai OCR error correction is proposed. The method combines the part-of-speech (POS) trigram model with a feature-based model. First, the POS trigram model is employed to correct non-word as well as real-word errors. In this step, the number of non-word errors are mostly reduced, but some real-word errors still remain because the POS trigram model cannot capture some useful features in discriminating candidate words. A feature-based approach using Winnow algorithm is then applied to correct the remaining errors. In order to overcome the expensive computation cost of the existing approach, we propose the idea of reducing the scope of correction by using word segmentation algorithm to find the approximate error strings from the input sentence. Though the word segmentation algorithm cannot give the accurate boundary of an error string, many of them can give clues of unknown strings which may be error strings. We can use this information to reduce the scope of correction from entire sentence to a more narrow scope. Next, to capture the characteristic of Thai OCR errors, we have defined the modified edit distance and use it to enumerate plausible candidates which deviate from the word in question within k-edit distance.

2 Problems of Thai OCR

The problem of OCR error correction can be defined as : given the string of characters $S = c_1c_2 \ldots c_n$ produced by OCR, find the word sequence $W = w_1w_2 \ldots w_l$ that maximizes the probability $P(W|S)$. Before describing the methods used to model $P(W|S)$, below we list some main characteristics of Thai that poses difficulties for correcting Thai OCR error.

- Words are written consecutively without word boundary delimiters such as white space characters. For example, the phrase "ญี่ปุ่นในปัจจุบัน" (Japan at present) in Figure 1, actually consists of three words: "ญี่ปุ่น" (Japan), "ใน" (at), and "ปัจจุบัน" (present). Therefore, Thai OCR error correction has to overcome word boundary ambiguity as well as select the most probable correction candidate at the same time. This is similar to the problem of *Connected Speech Recognition* and is sometimes called *Connected Text Recognition* (Ingels, 1996).
- There are 3 levels for placing Thai characters and some characters can occupy more than one level. For example, in Figure 2 "ฟุ้ง" consists of characters in three levels, i.e., ้, ุ, ง and ฟ are in the top, the bottom, the middle and both the middle and top levels, respectively. The character that occupies more than one level like ฟ usually connects to other characters (ฟ้) and causes error on the output of OCR, i.e., ฟ้ may be recognized as ฟ or โ. Therefore, to correct characters produced by OCR, not only substitution errors but also deletion and insertion errors must be considered. In addition, in such a case, the candidates ranked by OCR output are unreliable and cannot be used to reduce search space. This is because the connected characters tend to have very different features from the original separated ones.

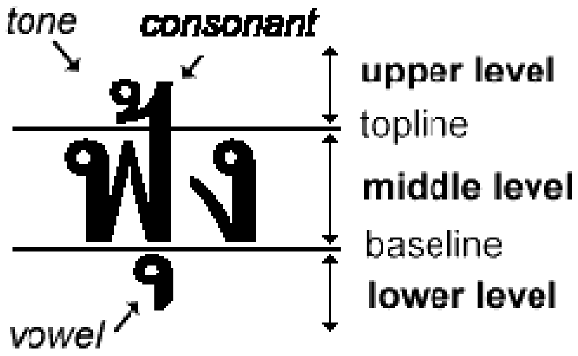

Figure 2: Three levels for placing Thai characters

3 Our Methods

3.1 Trigram Model

To find W that maximizes $P(W|S)$, we can use the POS trigram model as follows.

$$\begin{aligned} & \arg\max_W P(W|S) \\ &= \arg\max_W P(W)P(S|W)/P(S) \quad (1) \\ &= \arg\max_W P(W)P(S|W) \quad (2) \end{aligned}$$

The probability $P(W)$ is given by the language model and can be estimated by the trigram model as:

$$P(W) = P(W,T) = \prod P(t_i|t_{i-2}, t_{i-1})P(w_i|t_i) \quad (3)$$

$P(S|W)$ is the characteristics of specific OCR, and can be estimated by collecting statistical information from original text and the text produced by OCR. We assume that given the original word sequence W composed of characters $v_1v_2 \ldots v_m$, OCR produces the sequence as string S $(= c_1c_2 \ldots c_n)$ by repeatedly applying the following operation: substitute a character with another; insert a character; or delete a character. Let S_i be the i-prefix of S that is formed by first character to the i-character of S $(= c_1c_2 \ldots c_i)$, and similarly W_j is the j-prefix of W $(= v_1v_2 \ldots v_j)$. Using dynamic programming technique, we can calculate $P(S|W)$ $(= P(S_n|W_m))$ by the following equation:

$$\begin{aligned} P(S_i|W_j) = & \; max(P(S_{i-1}|W_j) * P(ins(c_i)), \\ & P(S_i|W_{j-1}) * P(del(v_j)), \\ & P(S_{i-1}|W_{j-1}) * P(c_i|v_j)) \quad (4) \end{aligned}$$

where $P(ins(c))$, $P(del(v))$ and $P(c|v)$ are the probabilities that letter c is inserted, letter v is deleted and letter v is substituted with c, respectively.

One method to do OCR error correction using the above model is to hypothesize all substrings in the input sentence as words (Nagata, 1996). Both words in the dictionary that exactly match with the substrings and those that approximately match are retrieved. To cope with unknown words, all other substrings not matched must also be considered. The word lattice is then scanned to find the N-best word sequences as correction candidates. In general, this method is perfectly good, except in one aspect: its time complexity. Because it generates a large number of hypothesized words and has to find the best combination among them, it is very slow.

3.2 Selective Trigram Model

To alleviate the above problem, we try to reduce the number of hypothesized words by generating them only when needed. Having analyzed the OCR output, we found that a large portion of input sentence are correctly recognized and need no approximation. Therefore, instead of hypothesizing blindly through the whole sentence, if we limit our hypotheses to only dubious areas, we can save considerable amount of time.

Following is our algorithm for correcting OCR output.

1. **Find dubious areas:** Find all substrings in the input sentence that exactly match words in the dictionary. Each substring may overlap with others. The remaining parts of sentence which are not covered by any of these substrings are considered as dubious areas.

2. **Make hypotheses for nonwords and unknown words:**

   (a) For each dubious string obtained from 1., the surrounding words are also considered to form candidates for correction by concatenating them with the dubious string. For example, in "*inform at j on*", j is an unknown string representing a dubious area, and *inform at* and *on* are words. In this case, the unknown word and its surrounding known words are combined together, resulting in "*informatjon*" as a new unknown string.

   (b) For each unknown string obtained form 2(a), apply the candidate generation routine to generate approximately matched words within k-edit distance. The value of k is varied proportionally to the length of candidate word.

   (c) All substrings except for ones that violate Thai spelling rules, i.e., lead by non-leading character, are hypothesized as unknown words.

3. **Find good word sequences:** Find the N-best word sequences according to equation (2). For unknown words, $P(w_i|\text{Unknown word})$ is computed by using the unknown word model in (Nagata, 1996).

4. **Make hypotheses for real-word error:** For each word w_i in N-best word sequence where the local probabilities $P(w_{i-1}, w_i, w_{i+1}, t_{i-1}, t_i, t_{i+1})$ are below a threshold, generate candidate words by applying the process similar to step 2 except that the nonword in step 2 is replaced with the word w_i. Find the word sequences whose probabilities computed by equation (2) are better than original ones.

5. **Find the N-best word sequences:** From all word sequences obtained from step 4, select the N-best ones.

The candidate generation routine uses a modification of the standard edit distance and employs the *error-tolerant finite-state recognition* algorithm (Oflazer, 1996) to generate candidate words. The modified edit distance allows arbitrary number of insertion and/or deletion of upper level and lower level characters, but allows no insertion or deletion of the middle level characters. In the middle level, it allows only k substitution. This is to reflect the characteristic of Thai OCR which, 1. tends to merge several characters into one when the character which spans two levels are adjacent to characters in the upper and lower level, and 2. rarely causes insertion and deletion errors in the middle level. For example, applying the candidate generation routine with 1 edit distance to the string "ฟุง" gives the set of candidates {ฟุ้ง, มุง, มุ่ง, มุ้ง, พุง, พุ่ง, ลุง, ซุง, ยุง, ยุ่ง, ยุ้ง}.

From our experiments, we found that the selective trigram model can deal with nonword errors fairly well. However, the model is not enough to correct real-word errors as well as words with the same part of speech. This is because the POS trigram model considers only coarse information of POS in a fixed restricted range of context, some useful information such as specific word collocation may be lost. Using word N-gram could recover some word-level information but requires an extremely large corpus to estimate all parameters accurately and consumes vast space resources to store the huge word N-gram table. In addition, the model losses generalized information at the level of POS.

For English, a number of methods have been proposed to cope with real-word errors in spelling correction (Golding, 1995; Golding and Roth, 1996; Golding and Schabes, 1993; Tong and Evans, 1996). Among them, the feature-based methods were shown to be superior to other approaches. This is because the methods can combine several kinds of features to determine the appropriate word in a given context. For our task, we adopt a feature-based algorithm called Winnow. There are two reasons why we select Winnow. First, it has been shown to be the best performer in English context-sensitive spelling correction (Golding and Roth, 1996). Second, it was shown to be able to handle difficult disambiguation tasks in Thai (Meknavin et al., 1997).

Below we describe Winnow algorithm that is used for correcting real-word error.

3.3 Winnow Algorithm

3.3.1 The algorithm

A Winnow algorithm used in our experiment is the algorithm described in (Blum, 1997). Winnow is a multiplicative weight updating and incremental algorithm (Littlestone, 1988; Golding and Roth, 1996). The algorithm is originally designed for learning two-class (positive and negative class) problems, and can be extended to multiple-class problems as shown in Figure 3.

Winnow can be viewed as a network of one target node connected to n nodes, called *specialists*, each of which examines one feature and

Let $v_1, \ldots, v_m$ be the values of the target concept to be learned, and x_i be the prediction of the i-specialist.

1. Initialize the weights $w_1, \ldots, w_n$ of all the specialists to 1.
2. **For Each** example $x = \{x_1, \ldots, x_n\}$ **Do**
   (a) Let V be the value of the target concept of the example.
   (b) Output $\hat{v}_j = \arg\max_{v_j \in \{v_1,\ldots,v_m\}} \sum_{i:x_i=v_j} w_i$
   (c) If the algorithm makes a mistake ($\hat{v}_j \neq V$), then:
      i. for each x_i equal to V, w_i is updated to $w_i \cdot \alpha$
      ii. for each x_i equal to $\hat{v}_j$, w_i is updated to $w_i \cdot \beta$

where, $\alpha > 1$ and $\beta < 1$ are promotion parameter and demotion parameter, and are set to 3/2 and 1/2, respectively.

Figure 3: The Winnow algorithm for learning multiple-class concept.

predicts x_i as the value of the target concept. The basic idea of the algorithm is that to extract some useful unknown features, the algorithm asks for opinions from all specialists, each of whom has his own specialty on one feature, and then makes a global prediction based on a weighted majority vote over all those opinions as described in Step 2-(a) of Figure 3. In our experiment, we have each specialist examine one or two attributes of an example. For example, a specialist may predict the value of the target concept by checking for the pairs "(attribute1 = value1) and (attribute2 = value2)". These pairs are candidates of features we are trying to extract.

A specialist only makes a prediction if its condition "(attribute1 = value1)" is true in case of one attribute, or both of its conditions "(attribute1 = value1) and (attibute2 = value2)" are true in case of two attributes, and in that case it predicts the most popular outcome out of the last k times it had the chance to predict. A specialist may choose to abstain instead of giving a prediction on any given example in case that it did not see the same value of an attribute in the example. In fact, we may have each specialist examines more than two attributes, but for the sake of simplification of preliminary experiment, let us assume that two attributes for each specialist are enough to learn the target concept.

The global algorithm updates the weight w_i of any specialist based on the vote of that specialist. The weight of any specialist is initialized to 1. In case that the global algorithm predicts incorrectly, the weight of the specialist that predicts incorrectly is halved and the weight of the specialist that predicts correctly is multiplied by 3/2. This weight updating method is the same as the one used in (Blum, 1997). The advantage of Winnow, which made us decide to use for our task, is that it is not sensitive to extra irrelevant features (Littlestone, 1988).

3.3.2 Constructing Confusion Set and Defining Features

To employ Winnow in correcting OCR errors, we first define *k-edit distance confusion set*. A k-edit distance confusion set $S = \{c, w_1, w_2, \ldots, w_n\}$ is composed of one centroid word c and words $w_1, w_2, \ldots, w_n$ generated by applying the candidate generation routine with maximum k modified edit distance to the centroid word. If a word c is produced by OCR output or by the previous step, then it may be corrected as $w_1, w_2, \ldots, w_n$ or c itself. For example, suppose that the centroid word is *know*, then all possible words in 1-edit distance confusion set are {*know, knob, knop, knot, knew, enow, snow, known, now*}. Furthermore, words with probability lower than a threshold are excluded from the set. For example, if a specific OCR has low probability of substituting t with w, "*knot*" should be excluded from the set.

Following previous works (Golding, 1995; Meknavin et al., 1997), we have tried two types of features: context words and collocations. Context-word features is used to test for the presence of a particular word within +/- M words of the target word, and collocations test for a pattern of up to L contiguous words and/or part-of-speech tags around the target word. In our experiment M and L is set to 10 and 2, respectively. Examples of features for discriminating between *snow* and *know* include:

(1) I {know, snow}
(2) winter within +10 words

where (1) is a collocation that tends to imply *know*, and (2) is a context-word that tends to imply *snow*. Then the algorithm should extract the features ("word within +10 words of the target word" = "winter") as well as ("one word before the target word" = "I") as useful features by assigning them with high weights.

3.3.3 Using the Network to Rank Sentences

After networks of k-edit distance confusion sets are learned by Winnow, the networks are used to correct the N-best sentences received from POS trigram model. For each sentence, every real word is evaluated by the network whose the centroid word is that real word. The network will then output the centroid word or any word in the confusion set according to the context. After the most probable word is determined, the confidence level of that word will be calculated. Since every specialist has weight voting for the target word, we can consider the weight as confidence level of that specialist for the word. We define the confidence level of any word as all weights that vote for that word divided by all weights in the network. Based on the confidence levels of all words in the sentence, the average of them is taken as the confidence level of the sentence. The N-best sentences are then reranked according to the confidence level of the sentences.

4 Experiments

We have prepared the corpus containing about 9,000 sentences (140,000 words, 1,300,000 characters) for evaluating our methods. The corpus is separated into two parts; the first part containing about 80 % of the whole corpus is used as a training set for both the trigram model and Winnow, and the rest is used as a test set. Based on the prepared corpus, experiments were conducted to compare our methods. The results are shown in Table 1, and Table 2.

Type	Error
Non-word Error	18.37%
Real-word Error	3.60%
Total	21.97%

Table 1: The percentage of word error from OCR

Type	Trigram	Trigram + Winnow
Non-word Error	82.16%	90.27%
Real-word Error	75.71%	87.60%
Introduced Error	1.42%	1.56%

Table 2: The percentage of corrected word errors after applying Trigram and Winnow

Table 1 shows the percentage of word errors from the entire text. Table 2 shows the percentage of corrected word errors after applying Trigram and Winnow. The result reveals that the trigram model can correct non-word and real-word, but introduced some new errors. By the trigram model, real-word errors are more difficult to correct than non-word. Combining Winnow to the trigram model, both types of errors are further reduced, and improvement of real-word error correction is more acute.

The reason for better performance of Trigram+Winnow over Trigram alone is that the former can exploit more useful features, i.e., context words and collocation features, in correction. For example, the word "นำ" (to bring) is frequently recognized as "น้ำ" (water) because the characters " ้ำ" is misreplaced with a single character " ำ" by OCR. In this case, Trigram cannot effectively recover the real-word error "นำ" to the correct word "น้ำ". The word "นำ" is effectively corrected by Winnow as the algorithm found the context words that indicate the occurence of "น้ำ" such as the words "ระเหย" (evaporate) and "พืช" (plant). Note that these context words cannot be used by Trigram to correct the real-word errors.

5 Conclusion

We have examined the application of the modified edit distance, POS trigram model and Winnow algorithm to the task of Thai OCR error correction. The experimental result shows that our proposed method reduces both non-word errors and real-word errors effectively. In future work, we plan to test the method with much more data and to incorporate other sources of information to improve the quality of correction. It is also interesting to examine how the method performs when applied to human-generated misspellings.

Acknowledgement

We would like to thank Paisarn Charoenpornsawat who helps us run experiment with Winnow. This work was partly supported by the Thai Government Research Fund.

References

Avrim Blum. 1997. Empirical support for winnow and weighted-majority algorithm: Results on a calendar scheduling domain. *Machine Learning*, 26.

Andrew R. Golding and Dan Roth. 1996. Applying winnow to context-sensitive spelling correction. In *Proceedings of the Thirteenth International Conference on Machine Learning.*

Andrew R. Golding and Yves Schabes. 1993. Combining trigram-based and featured-based methods for context-sensitive spelling correction. Technical Report TR-93-03a, Mitsubishi Electric Research Laboratory.

Andrew R. Golding. 1995. A bayesian hybrid method for context-sensitive spelling correction. In *Proceedings of the Third Workshop on Very Large Corpora.*

Peter Ingels. 1996. Connected text recognition using layered HMMs and token passing. In *Proceedings of the Second Conference on New Methods in Language Processing.*

Karen Kukich. 1992. Techniques for automatically correction words in text. *ACM Computing Surveys*, 24(4).

Nick Littlestone. 1988. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2.

Surapant Meknavin, Paisarn Charoenpornsawat, and Boonserm Kijsirikul. 1997. Feature-based Thai word segmentation. In *Proceedings of Natural Language Processing Pacific Rim Symposium '97.*

Masaaki Nagata. 1996. Context-base spelling correction for Japanese OCR. In *Proceedings of COLING '96.*

Kemal Oflazer. 1996. Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics*, 22(1).

Xiang Tong and David A. Evans. 1996. A statistical approach to automatic OCR error correction in context. In *Proceedings of the Fourth Workshop on Very Large Corpora.*