Error Correcting Romaji-kana Conversion for Japanese Language Education

Seiji Kasahara† **Mamoru Komachi**† **Masaaki Nagata**†† **Yuji Matsumoto**†

† Nara Institute of Science and Technology
8916-5 Takayama-cho, Ikoma-shi,
Nara, 630-0192 Japan
{seiji-k, komachi, matsu}@is.naist.jp

†† NTT Communication Science Laboratories
2-4 Hikari-dai, Seika-cho,
Soraku-gun,Kyoto, 619-0237 Japan
nagata.masaaki@lab.ntt.co.jp

Abstract

We present an approach to help editors of Japanese on a language learning SNS correct learners' sentences written in Roman characters by converting them into kana. Our system detects foreign words and converts only Japanese words even if they contain spelling errors. Experimental results show that our system achieves about 10 points higher conversion accuracy than traditional input method (IM). Error analysis reveals some tendencies of the errors specific to language learners.

1 Introduction

The Japan Foundation reports that more than 3.65 million people in 133 countries and regions were studying Japanese in 2009. Japanese is normally written in thousands of ideographic characters imported from Chinese (*kanji*) and about 50 unique syllabic scripts (*kana*). Because memorizing these characters is tough for people speaking European languages, many learners begin their study with *romaji*, or romanization of Japanese.

However, sentences written in kana are easier to edit for native Japanese than the ones in Roman characters. Converting Roman characters into kana helps Japanese editors correct learners' sentences, but naive romaji-kana conversion does not work well because there are spelling errors in learners' sentences. Even though traditional input methods have functionality to convert Roman characters into kana, existing IMs cannot treat learners' errors correctly since they are mainly designed for native Japanese speakers.

In this paper, we present an attempt to make the learner's sentences easier to read and correct for a native Japanese editor by converting erroneous text written in Roman characters into correct text written in kana while leaving foreign words unchanged. Our method consists of three steps: identification of language, spelling correction and converting text from Roman to kana. First, learners often write a word from their native language directly in a Japanese sentence. However, they are not converted correctly into their kana counterpart since the original spelling is usually not equivalent to the Japanese transliteration. Thus it is better to leave these word unchanged for the readability of editors. Second, since erroneous words cannot be converted correctly, spelling correction is effective. We combined filtering with cosine similarities and edit distance to correct learners' spelling errors. Third, we greedily convert Roman characters to kana for manual correction by native Japanese teachers. We compared our proposed system with a standard IM and conducted error analysis of our system, showing the characteristics of the learner's errors.

2 Related Work

Our interest is mainly focused on how to deal with erroneous inputs. Error detection and correction on sentences written in kana with kana character N-gram was proposed in (Shinnou, 1999). Our approach is similar to this, but our target is sentences in Roman characters and has the additional difficulty of language identification. Error-tolerant Chinese input methods were introduced in (Zheng et al., 2011; Chen and Lee, 2000). Though Roman-to-kana conversion is similar to pinyin-to-Chinese conversion, our target differs from them because our motivation is to help Japanese language teachers. Japanese commercial IMs such as Microsoft Office IME[1], ATOK[2], and Google IME[3] have a module of spelling correction, but their target is native Japanese speakers. (Ehara and Tanaka-Ishii, 2008) presented a high accuracy language detection system for text input. We perform

[1] http://www.microsoft.com/japan/office/2010/ime/default.mspx
[2] http://www.atok.com/
[3] http://www.google.com/intl/ja/ime/

Proceedings of the Workshop on Advances in Text Input Methods (WTIM 2011), pages 38–42,
Chiang Mai, Thailand, November 13, 2011.

error correction in addition to language identification. Correcting Japanese learners' error is also proposed in (Mizumoto et al., 2011). They try to correct sentences written in kana and kanji mixed, whereas we aim at texts in Roman characters.

3 Romanization of Japanese

There are some different standards of romanization in Japanese. The three main ones are Hepburn romanization, Kunrei-shiki Romaji, and Nihon-shiki Romaji. Most Japanese learners write in the Hepburn system, so we use this standard for our conversion system. Hepburn romanization generally follows English phonology with Romance vowels. It is an intuitive method of showing the pronunciation of a word in Japanese. The most common variant is to omit the macrons or circumflexes used to indicate a long vowel.

4 Romanized Japanese Learners Corpus from Lang-8

To our knowledge, there are no Japanese learners' copora written in Roman characters. Therefore, we collected text for a romanized Japanese learners' corpus from Lang-8[4], a language learning SNS. Since it does not officially distribute the data, we crawled the site in Dec 2010. It has approximately 75,000 users writing on a wide range of topics. There are 925,588 sentences written by Japanese learners and 763,971 (93.4%) are revised by human editors (Mizumoto et al., 2011). About 10,000 sentences of them are written in Roman characters. Table 1 shows some examples of sentences in Lang-8. As a feature of learners' sentences in Roman characters, most of them have delimiters between words, but verbs and their conjugational endings are conjoined. Another is the ambiguity of particle spelling. For example, "は" (topic marker) is assigned to *ha* by the conversion rule of Hepburn romanization, but it is pronounced as *wa*, so both of them are found in the corpus. Pairs of "を" *wo* (accusative case marker) and *o*, "へ" *he* (locative-goal case marker) and *e* also have the same ambiguity.

5 Error Tolerant Romaji-kana Conversion System

The system consists of three components: language identification, error correction with approximate matching, and Roman-to-kana conversion.

[4] http://lang-8.com/

5.1 Language Identification

Language identification is done by exact matching input sequences in English with a romanized[5] Japanese dictionary. Learners sometimes directly write words in their native language without adapting to Japanese romaji style. Since we are not focusing on implementing full transliteration (Knight and Graehl, 1998), we would like to convert only Japanese words into kana. To achieve this, we use an English word dictionary because most foreign words found in learners' sentences are English words. By adding dictionary, we can easily extend our system to another language. Those words matched with the dictionary are not converted. WordNet 2.1[6] is used as the dictionary. It has 155,287 unique words.

We also use a Japanese word dictionary to decide whether a word goes to the approximate word matching phase or not. The Japanese word dictionary is IPADic 2.7.0. We also use a dictionary of Japanese verb conjugations, because verbs in learners' sentence are followed by conjugational endings but they are separated in our word dictionary. The conjugation dictionary is made of all the occurrences of verbs and their conjugations extracted from Mainichi newspaper of 1991, with a Japanese dependency parser CaboCha 0.53[7] to find *bunsetsu* (phrase) containing at least one verb. The number of extracted unique conjugations is 243,663.

5.2 Error Correction

Words which are not matched in either the English or the Japanese dictionary in the language identification step are corrected by the following method. Spelling error correction is implemented by approximate word matching with two different measures. One is the cosine similarity of character unigrams. The other is edit distance. We use only IPADic to get approximate words.

5.2.1 Candidate Generation with Approximate Word Matching

First, we would like to select candidates with the minimum edit distance (Wagner and Fischer, 1974). Edit distance is the minimum number of editing operations (insertion, deletion and substitution) required to transform one string into an-

[5] Romanization was performed by kakasi 2.3.4. http://kakasi.namazu.org/

[6] http://wordnet.princeton.edu/

[7] http://chasen.org/~taku/software/cabocha/

Table 1: Examples of learners' sentences in Lang-8. Spell errors are underlined.

learners' sentence	correct	kana
yorushiku onegia shimasu.	yoroshiku onegai shimasu.	よろしく おねがい します。
Muscle musical wo mietai.	Muscle musical wo mitai.	Muscle musical を みたい。
anatah wa aigo ga wakarimasu ka.	anata wa eigo ga wakarimasu ka.	あなた は えいご が わかります か。

other. However, the computational cost of edit distance calculations can be a problem with a large vocabulary.[8] Therefore, we reduce the number of candidates using approximate word matching with cosine distance before calculating edit distance (Kukich, 1992). Cosine distance is calculated using character n-gram features. We set $n = 1$ because it covers most candidates in dictionary and reduces the number of candidates appropriately. For example, when we retrieved the approximate words for *packu* in our dictionary with cosine distance, the number of candidates is reduced to 163, and examples of retrieved words are *kau, pakku, chikau, pachikuri, etc*. Approximate word matching with cosine similarity can be performed very efficiently (Okazaki and Tsujii, 2010)[9] to get candidates from a large scale word dictionary.

5.2.2 Selecting the Most Likely Candidate

The system selects the correct word by choosing the most likely candidate by N-gram cost normalized by a word length. It is calculated with a romanized character 5-gram model built from kakasi-romanized Mainichi newspaper corpora of 1991 using SRILM 1.5.12[10] with Witten-Bell smoothing.[11]

5.3 Converting Roman Characters into Kana

We greedily convert Roman characters into the longest match kana characters. If a word includes character with circumflex, it is assumed to be two vowels meaning long sound (e.g., "kyôdai" is expanded as *kyoudai*: brother). Characters not used in the Hepburn system are assumed to be another character which has similar sound in English if possible. For example, *ca, ci, cu, ce ,co* are treated as *ka, shi, ku, se, ko* respectively.

Most kanas correspond to a pair of a consonant and a vowel. Although most pairs of Roman characters are converted into kana unambiguously, some pairs have several possibilities. One of them is a pair of *n* and following characters. For example, we can read Japanese word kinyuu as "きんゆう/*kin-yuu*: finance" and "きにゅう/*kinyuu*: entry." The reason why it occurs is that *n* can be a syllable alone. Solving this kind of ambiguity is out of scope of this paper; and we hope it is not a problem in practice, because after manual correction we can translate kana back to Roman characters unambiguously.

Table 2: Examples of successfully corrected word

misspelled	kana	correct	kana
shuutmatsu	しゅうｔまつ	shuumatsu	しゅうまつ
do-yoobi	どよおび	doyoubi	どようび
packu	ぱｃく	pakku	ぱっく

6 Experiments

We have evaluated our approach in converting Roman characters into kana after spelling error correction of sentences.

6.1 Evaluation Metrics

We evaluate the accuracy of word error correction. We also evaluate error correction performance with recall and precision. Recall and Precision are defined as follows:

$$Recall = N_t/N_w, Precision = N_t/N_e$$

where N_t, N_w and N_e denote the number of words corrected from wrong word to right word by the system, the number of words that contain errors, and the number of words edited by the system.

6.2 Experimental Settings

For comparison, we use Anthy 7900[12] as a baseline, which is one of the de facto standard open source IMs. It does not use either language identification or approximate word matching. Note that Anthy is not particularly tailored for spelling error correction. To compare with another system which has error correction function, we experimented with Google CGI API for Japanese Input[13]. Since it does not have Romaji-kana conversion module, the experiment was conducted using

[8] We set the maximum distance between input and candidate as 1, because it achieved the best accuracy in preliminary experiment.

[9] http://www.chokkan.org/software/simstring/

[10] http://www-speech.sri.com/projects/srilm/

[11] Witten-Bell smoothing works well compared to Kneser-Ney when data is very sparse.

[12] http://anthy.sourceforge.jp/

[13] http://www.google.com/intl/ja/ime/cgiapi.html

Table 3: Examples of uncorrected word

misspelled	kana	correct
renshou	れんしょう	renshuu
musugashi	むすがし	muzukashii
noryoukushiken	のりょうくしけん	nouryokushiken

Table 4: Performance of error correction

method	Acc	P	R
Anthy (baseline)	74.5	66.7	69.7
Anthy w/ Google API	77.8	69.8	72.9
Proposed w/o word match	84.5	76.6	77.3
Proposed w/ word match	85.0	78.1	78.6

Romaji-kana conversion by Anthy and error correction by Google API. We also compare our system with and without approximate word matching.

6.3 Data Set

We collected 500 sentences written in Roman characters from Lang-8. Although some of them have been already revised, we manually reannotated gold standard answers to enhance the consistency of quality. While making the test set, we corrected only spellings even if they contain other type of error because our main purpose is correcting spelling errors.[14]

6.4 Experimental Results

Table 4 shows the spelling correction accuracy. The word accuracy of the proposed system is 85.0% which is about 10 points higher than Anthy's 74.5%. The accuracy of our method without approximate word matching is 84.5%, showing that language identification is the crucial component of our method.[15] Examples of successfully corrected word are shown in Table 2. Underlined words are erroneous words and words underlined with wavy line are foreign words. Spelling correction with approximate matching can improve precision without degrading recall. However, the low performance of the baseline system shows difficulty of this task.

7 Discussion

Examples of uncorrected words are shown in Table 3. The top three largest ones are matching with valid word (40%), too large edit distance between original word and correct word (24%), and compound words (14%).

Matching with valid word: Matching with valid word occurs when the input matches a word in the dictionary. For example, if a learner incorrectly writes *renshou* instead of *renshuu*, it is not corrected because it is found in Japanese dictionary. This type of error cannot be corrected without context information so a word based language model is worth trying.

Table 5: Error types and system performance (percentage)

error type	number	corrected
Typo	31 (13.1)	7 (22.6)
Due to L1 phonetics	62 (26.3)	4 (6.5)
Due to L1 writing	28 (11.9)	2 (7.1)
Confusing vowels	88 (37.3)	7 (8.0)
Others	27 (11.4)	0.0 (0.0)
Total	236	20 (8.5)

Too large edit distance: A word whose edit distance from the input is larger than the threshold is not selected as a candidate. For example, if the learner writes *muzukashii* as *musugashi*, the edit distance between words is 3 which is lower than our threshold (=1). We can vary threshold but setting larger threshold introduces dissimilar words into the candidate list. Table 5 shows error types with their percentage against all erroneous words and system accuracy (where L1 means learners native language). Learners tend to confuse vowels and write erroneous word such as *domou* instead of *doumo*. Setting lower cost to edit operations of vowels than those of consonants may fix these kind of phonetic errors. A Japanese IM which lets us input kana and kanji by typing only consonants (Tanaka-Ishii et al., 2001) can be seen as a special case where the cost of edit operations of vowels is set to zero.

Compound words: Our system is effective when our dictionary and the learners' sentence use the same granularity of tokenization. For example, "*nouryokushiken*: capacity test" can be treated as two words, "*nouryoku*: capacity" and "*shiken*: test." In fact, IPADic does not have an entry for "*nouryoku shiken*." Therefore, the single word "*nouryokushiken*" does not hit when matching. To solve this problem, word segmentation techniques may be effective.

Acknowledgment

We would like to express our gratitude to our colleagues, Tomoya Mizumoto and Joseph Irwin for their cooperation.

[14] There are 3,274 words in the test data and 32 characters in a sentence on average.

[15] The number of foreign words in the test data is 137 and 124 words of them were correctly identified.

References

Zheng Chen and Kai-Fu Lee. 2000. A New Statistical Approach to Chinese Pinyin Input. In *Proceedings of ACL*, pages 241–247.

Yo Ehara and Kumiko Tanaka-Ishii. 2008. Multilingual Text Entry using Automatic Language Detection. In *Proceedings of IJCNLP*, pages 441–448.

Kevin Knight and Jonathan Graehl. 1998. Machine Transliteration. *Computational Linguistics*, 24(4):599–612.

Karen Kukich. 1992. Techniques for Automatically Correcting Words in Text. *ACM Computing Surveys*, 24(4):377–439.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining Revision Log of Language Learning SNS for Automated Japanese Error Correction of Second Language Learners. In *Proceedings of IJCNLP*.

Naoaki Okazaki and Jun'ichi Tsujii. 2010. Simple and Efficient Algorithm for Approximate Dictionary Matching. In *Proceedings of COLING*, pages 851–859.

Hiroyuki Shinnou. 1999. Detection and Correction for Errors in Hiragana Sequences by a Hiragana Character N-gram (in Japanese). *Transaction of Information Processing Society of Japan*, 40(6):2690–2698.

Kumiko Tanaka-Ishii, Yusuke Inutsuka, and Masato Takeichi. 2001. Japanese input system with digits –Can Japanese be input only with consonants? In *Proceedings of HLT*, pages 211–218.

Robert A. Wagner and Michael J. Fischer. 1974. The String to String Correction Problem. *Journal of the ACM*, 21(1):168–173.

Yabin Zheng, Chen Li, and Maosong Sun. 2011. CHIME: An Efficient Error-Tolerant Chinese Pinyin Input Method. In *Proceedings of IJCAI*, pages 2551–2556.