Detecting Transliterated Orthographic Variants via Two Similarity Metrics

Kiyonori Ohtake
ATR SLT
Keihanna Science City
Kyoto 619-0288,
Japan
kiyonori.ohtake@atr.jp

Youichi Sekiguchi
Nagaoka Univ. of Tech.
Nagaoka City,
Niigata 940-2188,
Japan
sekiguti@nlp.nagaokaut.ac.jp

Kazuhide Yamamoto
Nagaoka Univ. of Tech.
Nagaoka City,
Niigata 940-2188,
Japan
yamamoto@fw.ipsj.or.jp

Abstract

We propose a detection method for orthographic variants caused by transliteration in a large corpus. The method employs two similarities. One is string similarity based on edit distance. The other is contextual similarity by a vector space model. Experimental results show that the method performed a 0.889 F-measure in an open test.

1 Introduction

This paper discusses a detection method for transliterated orthographic variants of foreign words. Transliteration of foreign words causes orthographic variants because there are several conditions required for transliterating. One may person transliterate to approximate pronunciation, whereas another one may conduct transliteration based on spelling. For example, the English word *report* can be transliterated into two Japanese words, "リポート (ripooto)" and "レポート (repooto)." The former "ripooto" is based on an approximation of its pronunciation, while "repooto" is transliterated from its spelling.

In addition, several source languages can be transliterated. For instance, the English word *virus* corresponds to the Japanese words: "ウイルス (uirusu)" from Latin, "ビールス (biirusu)" and "ヴィールス (viirusu)" from German, while "バイラス (bairasu)" or "ヴァイラス (vairasu)" are also possible as transliterations that approximate the English pronunciation. Moreover, some foreign words end up in different forms in Japanese because of variation in English pronunciation; e.g., between British and American. For example, the English word *body* corresponds to two words: "ボディ (bodi)" from British and "バディ (badi)" from American.

One may think that if back-transliteration were done precisely, those variants would be back-transliterated into one word, and they would be recognized as variants. However, back-transliteration is known to be a very difficult task(Knight and Graehl, 1997).

Not only Japanese but any language that has a phonetic spelling has this problem of transliterated orthographic variants. For example, English has variants for a Chinese proper noun as "Shanhaiguan," "Shanhaikwan," or "Shanhaikuan."

Nowadays, it is well recognized that orthographic variant correction is an important processing step for achieving high performance in natural language processing. In order to achieve robust and reliable processing, we have to use many language resources: many types of corpora, dictionaries, thesauri, and so on. Orthographic variants cause many mismatches at any stage of natural language processing. In addition, not only orthographic variants, but also misspelled words tend to slip into a corpus. These words boost the perplexity of the corpus, and worsen the data sparseness problem.

To date, several studies have tried to cope with this orthographic variant problem; however, they considered the problem in a relatively clean corpus that was well organized by natives of the target language. As with orthographic variants, misspelled words cause mismatches, and we have to detect not only predictable orthographic variants but also misspelled variants. In addition, it is very hard to detect orthographic variants caused by misspelling with ordinary rule-based methods, because preparing such rules for misspellings that might be written is an unrealistic approach.

If a corpus includes texts that were written by non-natives of the language, orthographic variants that are misspelled will likely be increased because non-natives have a limited vocabulary in that language.

We propose a robust detection method for transliterated orthographic variants in a Japanese corpus. The method is marked by a combination of different types of similarities. It is not such a difficult task to detect simple misspelled words, because a large dictionary would tell us whether the word is common as long as we prepare a large enough dictionary. However, it often occurs that a misspelled word is recognized as a common word. For example, in English, someone may mistype "from" as "form," and string information will tell us nothing because both words are common. Therefore, we use contextual information to detect this kind of mistyping.

2 Transliteration for foreign words in Japanese: katakana

Japanese features three types of characters (katakana, hiragana, and kanji (Chinese characters)). Katakana is a syllabary which is used mostly to write Western loanwords, onomatopoeic words, names of plants and animals, non-Japanese personal and place names, for emphasis, and for slang, while hiragana is an ordinary syllabary.

Katakana cannot express the precise pronunciation of loanwords, because the katakana transliteration of a loanword is an attempt to approximate the pronunciation of its etymon (the foreign word from which it is derived). Thus, katakana orthography is often irregular, thus the same word may be written in multiple ways. Although there are general guidelines for loanword orthography, in practice there is considerable variation. In addition, recent years have seen an enormous increase in katakana use, not only in technical terminology, but in common daily usage.

To date, several detecting methods have been proposed for English and other languages. One may think that such methods can be applied to Japanese and work well. However, most katakana characters correspond to two phonemes, and this causes several problems. Due to the correspondence between katakana characters and phonemes, it is easy to imagine that the application would require tangled procedures.

2.1 Romanization

We use Japanese romanization for katakana characters to capture its pronunciation because there are several katakana characters for which the pronunciation is the same. For example, "ジ (possible romanization: zi/ji)" and "ヂ (possible romanization: di/zi)" are not differentiated in pronunciation. In addition, there are several katakana expressions that have very similar pronunciations. For instance, "チ (possible romanization: chi/ti)" in "チケット (*ticket*)" and "ティ (t'i)" in "ティケット (*ticket*)" are similar in pronunciation and they cause these variants. Naturally we can use katakana characters to compare two strings, but employing katakana characters makes the comparing procedure cumbersome and complicated. To avoid the complicated comparing procedure for katakana expressions, we use Japanese romanization.

We used a system of Japanese romanization based on ISO 3602, the romanization of which is based on the Kunreisiki system. There are two major systems for Japanese romanization. One is based on the theory of the Kunrei (Kunreisiki) system. The other one is the Hepburn system, which is widely used in English-speaking communities. The Kunreisiki system was designed to represent kana morphology accurately. For example, the katakana character "シ" is written as "si" in the Kunreisiki system while it is written as "shi" in the Hepburn system. In this example, the character "h" that is inserted disturbs simple matching procedures, because most katakana characters correspond to two romanized characters: a consonant and a vowel. Thus we prefer to use a romanization system based on the Kunreisiki system to make the matching procedure simple.

3 Detecting method

We propose a detecting method for katakana variants. The method consists of two components: one is string similarity and the other is contextual similarity.

The string similarity part measures similarity based on edit distance for katakana words. More precisely, there are two metrics of similarity: one measures the similarity between romanized strings of two words, while the other measures the similarity between raw strings of two words. We cannot use the romanization system as a perfect substitution of katakana, because romanization causes side effects. For example, both Japanese words "パ

ンチ (*punch*)" and "パンティ (*panty*)" are transliterated into "panti" by our romanization system. Thus, we use two string similarities.

Contextual similarity is defined as the distance between context vectors. A context vector we employed is marked by using a dependency structure of a sentence. A context vector is constructed by gathering surrounding information for the target katakana word, such as cooccurring nouns, the predicate expression depended upon by the katakana word, the particle marking the katakana word, and so on.

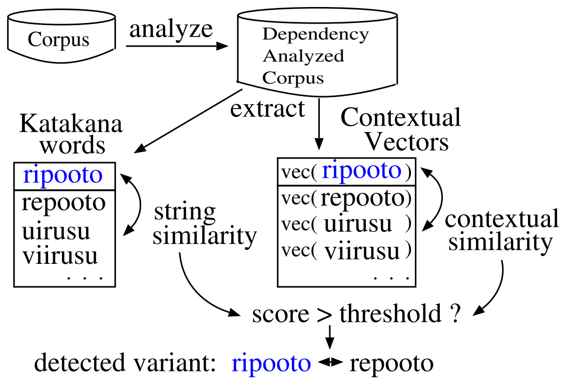


Figure 1: Overview of detecting katakana variants

Figure 1 shows an overview of the detecting method. The detection procedure is as follows:

1. Extract katakana words and contextual vectors from the dependency-analyzed result of the target corpus.
2. Choose a katakana word as the input word from the extracted katakana words.
3. Retrieve candidates of katakana variants from the extracted katakana words. Each candidate should share at least one character with the input word.
4. Calculate the similarity sim_{ed}, which is based on the ordinary edit distance, between the input Str_1 and each candidate Str_2. The similarity sim_{ed} is defined as follows:

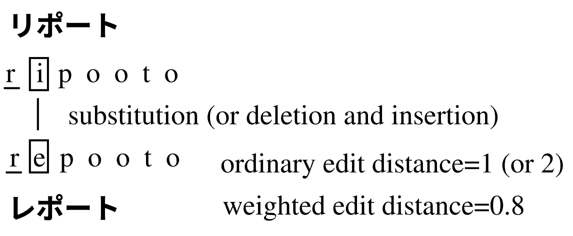
$$sim_{ed}(Str_1, Str_2) = 1 - \frac{ED(Str_1, Str_2)}{|Str_1| + |Str_2|}, \quad (1)$$

   where $ED(Str_1, Str_2)$ denotes the ordinary edit distance. If the input and a candidate word share suffix or prefix morphemes, the shared morphemes would be excluded from the comparing strings.
5. Calculate string similarity sim_s between the input and each candidate. If the input and a candidate word share a suffix or prefix morphemes, the shared morphemes would be excluded in the same way as above.
6. Calculate the contextual similarity sim_c between the input and each candidate.
7. Decide whether the candidate is a variant by means of a deciding module. The deciding module follows the decision list showed in Table 1, where we used the Gakken Kokugo Daijiten as the dictionary. It has almost 8,000 katakana words, and we slightly modified it.

We explain the details of the string similarity part and the contextual similarity part in the following subsections.

3.1 String similarity for romanized words

There are recognizable patterns in Japanese transliterated orthographic variants, thus, so far several rule-based methods to detect such variants have been developed. We use a kind of weighted edit distance to recognize transliterated orthographic variants. The weighting rules are very similar to the rules that are used in conventional rule-based methods. The ordinary edit distance between two strings is defined as the number of edit operations (insertion, deletion, and substitution, although sometimes substitution is not permitted) required to edit from one string into the other. Thus, the ordinary edit distance is a positive integer value. We defined small weighted operations in specific situations to identify recognizable patterns. Figure 2 shows an example of the difference between ordinary edit distance and weighted edit distance, in which we used a rule for changing a vowel that follows the same consonant ('r').



リポート
r i p o o t o
substitution (or deletion and insertion)
r e p o o t o
ordinary edit distance=1 (or 2)
レポート
weighted edit distance=0.8

Figure 2: Example of ordinary edit distance and weighted edit distance

More precisely, string similarity based on the weighted edit distance of romanizations for

length	frequency	sim_{ed}	sim_s	sim_c	decision
$> TH_{len}$	$*$	$> TH_{ed1}$	$> TH_{st1}$	$*$	variant
$<= TH_{len}$	$> TH_{freq}$	$*$	$*$	$< TH_{cos1}$	not variant
$< TH_{len}$	$*$	$*$	$*$	$> TH_{cos2}$	variant
Both words have entries in pre-defined dictionary					not variant
$*$	$*$	$> TH_{ed2}$	$> TH_{st2}$	$*$	variant
$*$	$*$	$*$	$*$	$*$	not variant

'$*$' means any conditions.

Table 1: Decision list of deciding module

katakana words A and B is defined as Formula (2),

$$Sim_s(A,B) = 1 - \frac{ED_k(rom(A), rom(B))}{|rom(A)| + |rom(B)|}, \tag{2}$$

where $rom(x)$ denotes romanized strings of x, and $ED_k(x,y)$ denotes a weighted edit distance between x and y that is specialized for katakana.

$ED_k(x,y)$ is a kind of weighted edit distance, and is marked by a distance function that determines the relaxed distance based on local strings. Here, $ED_k(x,y)$ is defined as Formula (3),

$$ED_k(x,y) = D(|x|,|y|), \tag{3}$$

where, for two strings S_1 and S_2, $D(i,j)$ is defined to be the specialized edit distance of $S_1[1..i]$ and $S_2[1..j]$.

$D(i,j)$ is given by the following recurrence relation:

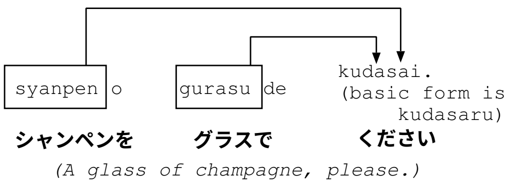
$$D(i,j) = \min \left[\begin{array}{l} D(i-1,j) + id(i,j), \\ D(i-1,j-1) + 2t(i,j), \\ D(i,j-1) + id(i,j) \end{array} \right], \tag{4}$$

where $id(i,j)$ defines the insertion and deletion operation distance, and that is defined to have the penalty value P_{id} if $S_1(i)$ or $S_2(j)$ denotes a consonant, and $id(i,j)$ has the value 1 in all other cases. In addition, $t(i,j)$ defines the substitution operation distance, and that is defined to have the value 0 if $S_1(i) = S_2(j)$, in all other cases, $t(i,j)$ has a pre-defined table and returns a value that depends on $S_1[i-3,..,i,..,i+3]$ and $S_2[j-3,..,j,..,j+3]$.

Table 2 shows an example of part of the $t(i,j)$ table. There are 29 entries in the $t(i,j)$ table. In Table 2, several $t(i,j)$ values are negative because in such a situation the strings compared have already had or will have a positive distance, thus the $t(i,j)$ has a negative value to adjust the distance.

3.2 Contextual similarity

In order to use the contextual information surrounding a katakana word, we employed a vector space model. We use a dependency analyzer to achieve more precise similarity, and contextual information is extracted from the dependency analyzed result of the text. Figure 3 shows an example of extracting vectors from an analyzed result, in which the vector has elements; N for cooccurring noun, P for predicate expression that is depended upon by the word, and PP for the particle and predicate expression pairs.



vec(syanpen) = [N; gurasu:1, P;kudasaru:1, PP;o-kudasaru : 1]

vec(gurasu) = [N; syanpen:1, P;kudasaru:1, PP;de-kudasaru : 1]

Figure 3: Extracting vector from dependency-analyzed result

The vectors are calculated by the following procedure.

1. Analyze the dependency structure for all sentences of the target corpus. We employed CaboCha[1] as the dependency analyzer.
2. Extract vectors for all katakana words included in the corpus. Each vector corresponds to a katakana word and consists of the following elements:
    - Nouns that cooccur with the katakana word.

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

$i-3$ $j-3$	$i-2$ $j-2$	$i-1$ $j-1$	i j	$i+1$ $j+1$	$i+2$ $j+2$	$i+3$ $j+3$	$t(i,j)$
$*$ $*$	$*$ $*$	$S_2(j-1)$ $S_1(i-1)$	y i	u $S_1(i+2)$	[kmnt] y	i u	-0.3
y $*$	u i	[kmnt] $S_1(i-1)$	i y	u u	$*$ $*$	$*$ $*$	-0.3
$*$ $*$	$*$ $*$	$*$ $*$	[dz] [dz]	i i	$*$ $*$	$*$ $*$	0.25

'$*$' means any character.
'[ ]' means character class in a regular expression.

Table 2: A part of $t(i,j)$ table

- Predicate that is depended upon by the katakana word.
- Particle and predicate pair: particle that follows the katakana word and predicate that is depended upon by the katakana word.

Each element is extracted from the dependency-analyzed result of a sentence, and the frequency of the element is counted.

3. Load a tf-idf-like weight onto each element of the vector. The weight is calculated by the following formula.

$$W(kw_i, e_i) = f(kw_i, e_i) \log \left(\frac{N}{sf(kw_i)} \right). \quad (5)$$

Here, kw_i is a katakana word, e_i is an element of the vector corresponding to kw_i, $f(kw_i, e_i)$ denotes the frequency of the element e_i for kw_i, $sf(kw_i)$ denotes the frequency of the sentence including kw_i, and N denotes the number of katakana words in the corpus.

The contextual similarity is defined as the following formula.

$$sim_c(kw_i, kw_j) = \cos(vec(kw_i), vec(kw_j))$$
$$= \frac{\sum_{e_x} W(kw_i, e_x) W(kw_j, e_x)}{\sqrt{\sum_{e_m} W(kw_i, e_m)^2} \sqrt{\sum_{e_n} W(kw_j, e_n)^2}}, \quad (6)$$

where $vec(kw)$ denotes a vector corresponding to the katakana word kw.

4 Experiments

We used the ATR Basic Travel Expression Corpus (BTEC)(Takezawa et al., 2002) as a resource for text. BTEC is a multilingual corpus and was mainly developed with English and Japanese. The Japanese part of BTEC contains not only ordinary katakana variants, but also mis-transliterated katakana strings by non-Japanese natives that serve as our target for detection. The BTEC we used consists of almost 200,000 sentences.

We used almost 160,000 sentences for the development of the $t(i,j)$ table, other rules used in our method, and parameter estimations for the method. We manually estimated the parameters to achieve the highest F-measure for the development sentences, and estimated the parameters as follows: $P_{id} = 2.5$, $TH_{len} = 5$, $TH_{st1} = 0.94$, $TH_{freq} = 3$, $TH_{cos1} = 0.12$, $TH_{cos2} = 0.02$, $TH_{ed} = 0.65$, and $TH_{st2} = 0.89$.

The developmental corpus includes almost 6,000 types of katakana words. We carried out a closed test using the development corpus with these parameter settings. There are two choices for the detection method: One is the use of a dictionary to judge whether the input and candidate words are known as different words. The other is the use of contextual similarity. Actually, in the detection method, contextual similarity plays a supportive role because there is a data sparseness problem. Therefore we carried out an experiment with four conditions. Table 3 shows the results of recall, precision, and F-measure on these four conditions.

The remaining 40,000 sentences were used as a test set, with which we carried out an open test. Table 4 shows the result of the open test.

There is an obvious tendency for the detection of short words to be very difficult. We compared an F-measure for each class of word length, with Figure 4 showing the results in open tests with the dictionary and without it. Both open tests were conducted without contextual similarity. Compar-

Recall	Precision	F
with dictionary, with context		
0.820 (296/361)	0.931 (296/318)	0.872
with dictionary, without context		
0.850 (307/361)	0.930 (307/330)	0.889
without dictionary, with context		
0.823 (297/361)	0.903 (297/329)	0.861
without dictionary, without context		
0.850 (307/361)	0.862 (307/356)	0.856

Table 3: Closed test results

Recall	Precision	F
with dictionary, with context		
0.827 (62/75)	0.886 (62/70)	0.855
with dictionary, without context		
0.907 (68/75)	0.872 (68/78)	0.889
without dictionary, with context		
0.800 (60/75)	0.822 (60/73)	0.811
without dictionary, without context		
0.880 (66/75)	0.725 (66/91)	0.795

Table 4: Open test results

ing these results tells us the effect of the introduced dictionary.

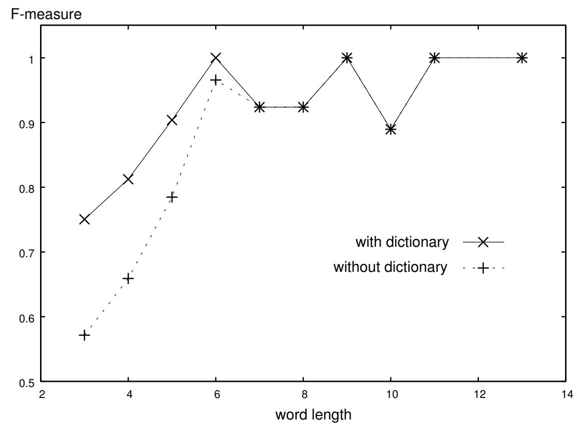


Figure 4: Results of open test with dictionary and without dictionary in each word length

5 Discussion

The experimental results showed that it is very difficult to detect short variants. Thus, it is reasonable to use a dictionary for words that we already know. Figure 4 shows the impact of the dictionary.

Most of the detection errors are related to proper nouns, because there are many proper nouns that are difficult to recognize as different words, such as "マリー (marii, *Mary*)" and "マリア (maria, *Maria*)," and so on. Furthermore, it is also hard to differentiate these words by their contextual vectors, because using proper nouns is independent of context. If we can precisely detect these proper nouns written in katakana, we will be able to avoid such mis-detection. In practical situations, an enormous dictionary of proper nouns such as ENAMDICT[2] would be useful for this problem.

The detection method successfully detected several mistyped words, such as "ブラン (buran)" for "ブラウン (buraun, *Brown*)," "フリーモントホテル (furiimonto hoteru)" for "フレモントホテル (furemonto hoteru, *the Fremont Hotel*)," and so on. Most of the detected mistypes were vowels, because the string similarity is designed to be tolerant of mistyped vowels, and such mistypes were detected successfully. However, it seems to be a difficult task to detect mistypes with consonants, because, in ordinary situations, most mistypes related to consonants seem to be completely different words.

In addition, there are several variants that are difficult to detect by this method. A typical example was shown in the Introduction: "ウイルス (uirusu)" or "ビールス (biirusu)" for the English word *virus*. This type of variant includes drastic changes; for instance, the ordinary edit distance between "ウイルス (uirusu)" and "ビールス (biirusu)" is four, and the similarity derived from the distance is too small (0.5) to identify it as the same word. Moreover, there is another type of orthographic variant that has changed with time. BTEC includes such an example: for the English word *milkshake*, both "ミルクセーキ (mirukuseeki)" and "ミルクシェーク (mirukusyeeku)" exist. We have to be careful of this problem when we process a corpus that has been developed for quite some time, and that includes both very old texts and new ones.

A well known problem arises here: data sparseness. Orthographic variants appear less frequently than their standard expressions, and we cannot expect to have much contextual information for orthographic variants. Therefore, we always have to cope with this problem even when we process very large corpus, because the appearance of variants does not relate to the size of the corpus. The

[2]`http://www.csse.monash.edu.au/~jwb/enamdict_doc.html`

basic idea of the contextual vector seems very reasonable for words that appear frequently in a target corpus. However, experimental results showed that the contextual similarity did not work as expected because of this data sparseness. Consequently, to achieve reliable contextual similarity, we have to use sentences in which a candidate of the variant is used. On-line WWW searching seems to be good as such a resource for variant detection because WWW texts include many variants.

On the other hand, there was a pair of words that have very high string similarity and contextual similarity, but they are not variants. That pair is "シャンペン (syanpen *champagne*): サインペン (sainpen *sign pen / felt-tip pen*)," and examples of sentences that include each word are as follows:

> syanpen o gurasu de kudasai. (*Give me a glass of champagne, please.*)
>
> sainpen o ippon kudasai. (*Give me a felt-tip pen, please.*)

Both words are arguments of the same verb "kudasai," and the vectors derived from the analyzed result of these sentences would be very similar. Practically, these words are identified as different words by using a dictionary. However, when using only contextual similarity, these words would be judged as variants.

It is not easy to detect all of the variants by applying the proposed method. Indeed, the method employs contextual information to achieve good performance, but the contextual information used also includes variants, and the variants cause mismatches. In addition, not only katakana variants, but also other orthographic variants, such as kanji and cross-script orthographic variants (e.g., kanji vs. hiragana, hiragana vs. katakana, and so on), should be detected to achieve high precision and recall.

6 Related works

To date, there have been several studies conducted on the detection of transliterated orthographic variants(e.g., (Kubota et al., 1994; Shishibori et al., 1994)). Most of these, however, targeted a relatively clean and well organized corpus or they assumed artificial situations. As a practical matter, not only predictable orthographic variants but also misspelled words should be detected. The detection of transliterated orthographic variants and spelling corrections have been studied separately, and there is no study that is directly related to our work.

There are several studies on transliteration (e.g., (Knight and Graehl, 1997)), and they tell us that machine transliteration of language pairs that employ very different alphabets and sound systems is extremely difficult, and that the technology is still to immature for use in practical processing.

7 Conclusion

We propose a method to detect transliterated orthographic variants. The method is marked by the use of string similarity and contextual similarity via contextual vectors. The method achieved a 0.889 F-measure in an open test. The results showed that detection of short word variants is very difficult, and a dictionary raised the precision for such short words. However, contextual similarity did not contribute as expected to the detection of orthographic variants.

Acknowledgements

This research was supported in part by the Ministry of Public Management, Home Affairs, Posts and Telecommunications. The authors thank the anonymous reviewers for their suggestive and encouraging comments.

References

Kevin Knight and Jonathan Graehl. 1997. Machine transliteration. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, pages 128–135.

Jun'ichi Kubota, Yukie Shoda, Masahiro Kawai, Hirofumi Tamagawa, and Ryoichi Sugimura. 1994. A method of detecting katakana variants in a document. *The Transaction of IPSJ*, 35(12):2745–2751. (in Japanese).

Masami Shishibori, Kazuhiko Tsuda, and Jun'ichi Aoe. 1994. A method for generation and normalization of katakana variant notations. *The Transactions of IEICE*, J-77-DII(2):380–387. (in Japanese).

Toshiyuki Takezawa, Eiichiro Sumita, Fumiaki Sugaya, Hirofumi Yamamoto, and Seiichi Yamamoto. 2002. Toward a broad-coverage bilingual corpus for speech translation of travel converstaions in the real world. In *Proceedings of LREC 2002*, volume 1, pages 147–152.