Splitting Noun Compounds via Monolingual and Bilingual Paraphrasing: A Study on Japanese Katakana Words

Nobuhiro Kaji
Institute of Industrial Science
University of Tokyo, Tokyo, Japan
kaji@tkl.iis.u-tokyo.ac.jp

Masaru Kitsuregawa
Institute of Industrial Science
University of Tokyo, Tokyo, Japan
kitsure@tkl.iis.u-tokyo.ac.jp

Abstract

Word boundaries within noun compounds are not marked by white spaces in a number of languages, unlike in English, and it is beneficial for various NLP applications to split such noun compounds. In the case of Japanese, noun compounds made up of katakana words (i.e., transliterated foreign words) are particularly difficult to split, because katakana words are highly productive and are often out-of-vocabulary. To overcome this difficulty, we propose using monolingual and bilingual paraphrases of katakana noun compounds for identifying word boundaries. Experiments demonstrated that splitting accuracy is substantially improved by extracting such paraphrases from unlabeled textual data, the Web in our case, and then using that information for constructing splitting models.

1 Introduction

1.1 Japanese katakana words and noun compound splitting

Borrowing is a major type of word formation in Japanese, and numerous foreign words (proper names or neologisms etc.) are continuously being imported from other languages (Tsujimura, 2006). Most borrowed words in modern Japanese are transliterations[1] from English and they are referred to as *katakana words* because transliterated foreign words are primarily spelled by using katakana characters in the Japanese writing system.[2] Compounding is another type of word formation that is common in Japanese (Tsujimura, 2006). In particular, noun compounds are frequently produced by merging two or more nouns together. These two types of word formation yield a significant amount of katakana noun compounds, making Japanese a highly productive language.

In Japanese as well as some European and Asian languages (e.g., German, Dutch and Korean), constituent words of compounds are not separated by white spaces, unlike in English. In those languages, it is beneficial for various NLP applications to split such compounds. For example, compound splitting enables SMT systems to translate a compound on a word-by-word basis, even if the compound itself is not found in the translation table (Koehn and Knight, 2003; Dyer, 2009). In the context of IR, decompounding has an analogous effect to stemming, and it significantly improves retrieval results (Braschler and Ripplinger, 2004). In abbreviation recognition, the definition of an abbreviation is often in the form of a noun compound, and most abbreviation recognition algorithms assume that the definition is properly segmented; see e.g., (Schwartz and Hearst, 2003; Okazaki et al., 2008).

This has led NLP researchers to explore methods for splitting compounds, especially noun compounds, in various languages (Koehn and Knight, 2003; Nakazawa et al., 2005; Alfonseca et al., 2008a). While many methods have been presented, they basically require expensive linguistic resources to achieve high enough accuracy. For example, Alfonseca et al. (2008b) employed a word dictionary, which is obviously useful for this task. Other studies have suggested using bilingual resources such as parallel corpora (Brown, 2002; Koehn and Knight,

[1] Some researchers use the term transcription rather than transliteration (Breen, 2009). Our terminology is based on studies on machine transliteration (Knight and Graehl, 1998).

[2] The Japanese writing system has four character types: hiragana, katakana, kanji, and Latin alphabet.

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 959–969,
Edinburgh, Scotland, UK, July 27–31, 2011.

2003; Nakazawa et al., 2005). The idea behind those methods is that compounds are basically split into constituent words when they are translated into English, where the compounded words are separated by white spaces, and hence splitting rules can be learned by discovering word alignments in bilingual resources.

The largest obstacle that makes compound splitting difficult is the existence of out-of-vocabulary words, which are not found in the abovementioned linguistic resources. In the Japanese case, it is known that katakana words constitute a large source of out-of-vocabulary words (Brill et al., 2001; Nakazawa et al., 2005; Breen, 2009). As we have discussed, katakana words are very productive, and thus we can no longer expect existent linguistic resources to have sufficient coverage. According to (Breen, 2009), as many as 20% of katakana words in news articles, which we think include less out-of-vocabulary words than Web and other noisy textual data, are out-of-vocabulary. Those katakana words often form noun compounds, and pose a substantial difficulty for Japanese text processing (Nakazawa et al., 2005).

1.2 Paraphrases as implicit word boundaries

To alleviate the errors caused by out-of-vocabulary words, we explored the use of unlabeled textual data for splitting katakana noun compounds. Since the amount of unlabeled text available is generally much larger than word dictionaries and other expensive linguistic resources, it is crucial to establish a methodology for taking full advantage of such easily available textual data. While several approaches have already been proposed, their accuracies are still unsatisfactory (section 2.1).

From a broad perspective, our approach can be seen as using paraphrases of noun compounds. As we will see in section 4 and 5, katakana noun compounds can be paraphrased into various forms that strongly indicate word boundaries within the original noun compound. This paper empirically demonstrates that splitting accuracy can be significantly improved by extracting such paraphrases from unlabeled text, the Web in our case, and then using that information for constructing splitting models.

Specifically, two types of paraphrases are investigated in this paper. Section 4 explores monolingual paraphrases that can be generated by inserting certain linguistic markers between constituent words of katakana noun compounds. Section 5, in turn, explores bilingual paraphrases (specifically, back-transliteration). Since katakana words are basically transliterations from English, back-transliterating katakana noun compounds is also useful for splitting. To avoid terminological confusion, monolingual paraphrases are simply referred to as paraphrases and bilingual paraphrases are referred to as back-transliterations hereafter.

We did experiments to empirically evaluate our method. The results demonstrated that both paraphrase and back-transliteration substantially improved the performance in terms of F_1-score, and the best performance was achieved when they were combined. We also confirmed that our method outperforms the previously proposed splitting methods by a wide margin. All these results strongly suggest the effectiveness of paraphrasing and back-transliteration for identifying word boundaries within katakana noun compounds.

2 Related Work

2.1 Compound splitting

A common approach to splitting compounds without expensive linguistic resources is an unsupervised method based on word or string frequencies estimated from unlabeled text (Koehn and Knight, 2003; Ando and Lee, 2003; Schiller, 2005; Nakazawa et al., 2005; Holz and Biemann, 2008). Amongst others, Nakazawa et al. (2005) also investigated ways of splitting katakana noun compounds. Although the frequency-based method generally achieves high recall, its precision is not satisfactory (Koehn and Knight, 2003; Nakazawa et al., 2005). Our experiments empirically compared our method with the frequency-based methods, and the results demonstrate the advantage of our method.

Our approach can be seen as augmenting discriminative models of compound splitting with large external linguistic resources, i.e., textual data on the Web. In a similar spirit, Alfonseca et al. (2008b) proposed the use of query logs for compound splitting.[3] Their experimental results, however, did not clearly

[3] Although they also proposed using anchor text, this slightly degraded the performance.

demonstrate their method's effectiveness. Without the query logs, the accuracy is reported to drop only slightly from 90.55% to 90.45%. In contrast, our experimental results showed statistically significant improvements as a result of using additional resources. Moreover, we used only textual data, which is easily available, unlike query logs.

Holz and Biemann (2008) proposed a method for splitting and paraphrasing German compounds. While their work is related to ours, their algorithm is a pipeline model and paraphrasing result is not employed during splitting.

2.2 Other research topics

Our study is closely related to word segmentation, which is an important research topic in Asian languages including Japanese. Although we can use existing word segmentation systems for splitting katakana noun compounds, it is difficult to reach the desired accuracy, as we will empirically demonstrate in section 6. One reason for this is that katakana noun compounds often include out-of-vocabulary words, which are difficult for the existing segmentation systems to deal with. See (Nakazawa et al., 2005) for a discussion of this point. From a word segmentation perspective, our task can be seen as a case study focusing on a certain linguistic phenomenon of particular difficulty. More importantly, we are unaware of any attempts to use paraphrases or transliterations for word segmentation in the same way as we do.

Recent studies have explored using paraphrase statistics for parsing (Nakov and Hearst, 2005a; Nakov and Hearst, 2005b; Bansal and Klein, 2011). Although these studies successfully demonstrated the usefulness of paraphrases for improving parsers, the connection between paraphrases and word segmentation (or noun compound splitting) was not at all discussed.

Our method of using back-transliterations for splitting katakana noun compounds (section 5) is closely related to methods for mining transliteration from the Web text (Brill et al., 2001; Cao et al., 2007; Oh and Isahara, 2008; Wu et al., 2009). What most differentiates these studies from our work is that their primary goal is to build a machine transliteration system or to build a bilingual dictionary itself; none of them explored splitting compounds.

Table 1: Basic features.

ID	Feature	Description
1	y_i	constituent word 1-gram
2	$y_{i-1}y_i$	constituent word 2-gram
3	LEN(y_i)	#characters of y_i (1, 2, 3, 4, or $\geq$5)
4	DICT(y_i)	true if y_i is in the dictionary

3 Supervised Approach

The task we examine in this paper is splitting a katakana noun compound x into its constituent words, $\boldsymbol{y} = (y_1, y_2 \ldots y_{|\boldsymbol{y}|})$. Note that the output can be a single word, i.e., $|\boldsymbol{y}| = 1$. Since it is possible that the input is an out-of-vocabulary word, it is not at all trivial to identify a single word as such. A naive method would erroneously split an out-of-vocabulary word into multiple constituent words.

We formalize our task as a structure prediction problem that, given a katakana noun compound x, predicts the most probable splitting $\boldsymbol{y}^*$.

$$\boldsymbol{y}^* = \underset{\boldsymbol{y} \in \mathcal{Y}(x)}{\operatorname{argmax}} \boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{y}),$$

where $\mathcal{Y}(x)$ represents the set of all splitting options of x, $\boldsymbol{\phi}(\boldsymbol{y})$ is a feature vector representation of $\boldsymbol{y}$, and $\boldsymbol{w}$ is a weight vector to be estimated from labeled data.

Table 1 summarizes our basic feature set. Features 1 and 2 are word 1-gram and 2-gram features, respectively. Feature 3 represents the length of the constituent word. LEN(y) returns the number of characters of y (1, 2, 3, 4, or $\geq$5). Feature 4 indicates whether the constituent word is registered in an external dictionary (see section 6.1). DICT(y) returns true if the word y is in the dictionary.

In addition to those basic features, we also employ paraphrases and back-transliterations of katakana noun compounds as features. The features are detailed in sections 4 and 5, respectively.

We can optimize the weight vector $\boldsymbol{w}$ using an arbitrary training algorithm. Here we adopt the averaged perceptron algorithm for the sake of time efficiency (Freund and Schapire, 1999). The perceptron offers efficient online training, and it performs comparatively well with batch algorithms such as SVMs. Since we use only factored features (see table 1, section 4 and section 5), dynamic programming can be used to locate $\boldsymbol{y}^*$.

Table 2: Paraphrase rules and examples. The first column represents the type of linguistic marker to be inserted, the second column shows the paraphrase rules, and the last column gives examples.

Type	Rule	Example
Centered dot	$X_1X_2 \rightarrow X_1 \cdot X_2$	アンチョビパスタ → アンチョビ・パスタ (anchovy pasta) (anchovy · pasta)
Possessive marker	$X_1X_2 \rightarrow X_1$ の X_2	アンチョビパスタ → アンチョビの パスタ (anchovy pasta) (with anchovy) (pasta)
Verbal suffix	$X_1X_2 \rightarrow X_1$ する X_2 $X_1X_2 \rightarrow X_1$ した X_2	ダウンロードファイル→ダウンロードしたファイル (download file) (downloaded) (file)
Adjectival suffix	$X_1X_2 \rightarrow X_1$ な X_2 $X_1X_2 \rightarrow X_1$ 的 X_2 $X_1X_2 \rightarrow X_1$ 的な X_2	サプライズギフト → サプライズなギフト (surprise gift) (surprising) (gift)

4 Paraphrasing

In this section, we argue that paraphrases of katakana noun compounds provides useful information on word boundaries. Consequently, we propose using paraphrase frequencies as features for training the discriminative model.

4.1 Paraphrasing noun compounds

A katakana noun compound can be paraphrased into various forms, some of which provide information on the word boundaries within the original compound.

(1) a. アンチョビパスタ
(anchovy pasta)
b. アンチョビ・パスタ
(anchovy · pasta)
c. アンチョビの パスタ
(with anchovy) (pasta)

These examples are paraphrases of each other. (1a) is in the form of a noun compound, within which the word boundary is ambiguous. In (1b), on the other hand, a centered dot ・ is inserted between the constituent words. In the Japanese writing system, the centered dot is sometimes, but not always, used to separate long katakana compounds for the sake of readability. (1c) is the noun phrase generated from (1a) by inserting the possessive marker 'の', which can be translated as *with* in this context, between the constituent words. If we observe paraphrases of (1a) such as (1b) and (1c), we can guess that a word boundary exists between 'アンチョビ (anchovy)' and 'パスタ (pasta)'.

4.2 Paraphrase rules

The above discussion led us to use paraphrase frequencies estimated from Web text for splitting katakana noun compounds. For this purpose, we established the seven paraphrase rules illustrated in Table 2. The rules are in the form of $X_1X_2 \rightarrow X_1MX_2$, where X_1 and X_2 represent nouns, and M is a certain linguistic marker (e.g., the possessive marker 'の'). The left-hand term corresponds to a compound to be paraphrased and the right-hand term represents its paraphrase. For instance, X_1 = 'アンチョビ (anchovy)', X_2 = 'パスタ (pasta)', and M = 'の'. The paraphrase rules we use are based on the rules proposed by Kageura et al. (2004) for expanding complex terms, primarily noun compounds, into their variants.

4.3 Web-based frequency as features

We introduce a new feature using the paraphrase rules and Web text. As preprocessing, we use regular expressions to count the frequencies of all potential paraphrases of katakana noun compounds on the Web in advance.

(katakana)+ ・ (katakana)+
(katakana)+ の (katakana)+
(katakana)+ する (katakana)+
...

where (katakana) corresponds to one katakana character. Given a candidate segmentation $\boldsymbol{y}$ at test time, we generate paraphrases of the noun compound by setting $X_1 = y_{i-1}$ and $X_2 = y_i$, and applying the paraphrase rules. We then use $\log(F + 1)$, where F is the sum of the Web-based frequencies of the gen-

erated paraphrases, as the feature of the boundary between y_{i-1} and y_i.

As the feature value, we use the logarithmic frequency, rather than the raw frequency, for scaling. Since the other features have binary value, we found, in initial experiments, that the importance of this feature is overemphasized if we use the raw frequency. Note that we use $\log(F+1)$ rather than $\log F$ so as to avoid the feature value being zero when $F = 1$.

5 Back-transliteration

Most katakana words are transliterations from English, where words are separated by white spaces. It is, therefore, reasonable to think that back-transliterating katakana noun compounds into English would provide information on word boundaries, in a similar way to paraphrasing.

This section presents a method for extracting back-transliterations of katakana words from monolingual Web text, and establishing word alignments between those katakana and English words (Table 3). In what follows, the pair of katakana words and its English back-transliteration is referred to as a *transliteration pair*. If the transliteration pair is annotated with word alignment information as in Table 3, it is referred to as a *word-aligned transliteration pair*.

Using word-aligned transliteration pairs extracted from the Web text, we derive a binary feature indicating whether katakana word y_i corresponds to a single English word. Additionally, we derive another feature indicating whether a katakana word 2-gram $y_{i-1}y_i$ corresponds to an English word 2-gram.

5.1 Parenthetical expressions

In Japanese and other Asian languages, transliterated words are sometimes followed by their English back-transliterations inside parentheses:

(2) a. アメリカでは ‖ ジャンク (junk) フード (food) (junk food) と...

b. トラックバック ‖ スパム (spam) (spam) を撃退する...

where the underline indicates the Japanese text that is followed by English back-transliteration. We extract word-aligned transliteration pairs from such parenthetical expressions by establishing the correspondences between pre-parenthesis and in-parenthesis words.

Table 3: Word-aligned transliteration pairs. The number indicates the word alignment.

Japanese	**English**
ジャンク$_1$フード$_2$	junk$_1$ food$_2$
スパム$_3$	spam$_3$

To accomplish this, we have to resolve three problems: (a) English words inside parenthesis do not always provide a back-transliteration of the pre-parenthesis text, (b) the left boundary of the pre-parenthesis text, denoted as '‖' in the example, has to be identified, and (c) pre-parenthesis text, which is a katakana noun compound in our case, has to be segmented into words.

Although several studies have explored mining transliterations from such parenthetical expressions (Cao et al., 2007; Wu et al., 2009), the last problem has not been given much attention. In the past studies, the pre-parenthesis text is assumed to be correctly segmented by, typically, using existent word segmentation systems. This is, however, not appropriate for our purpose, because pre-parenthesis text is a katakana noun compound, which is hard for existing systems to handle, and hence the alignment quality is inevitably affected by segmentation errors.

To handle these three problems, we use the phonetic properties of the transliterations. For the purpose of explanation, we shall first focus on problem (c). Since transliterated katakana words preserve the pronunciation of the original English words to some extent (Knight and Graehl, 1998), we can discover the correspondences between substrings of the two languages based on phonetic similarity:

(3) a. [ジャン]$_1$[ク]$_2$[フー]$_3$[ド]$_4$

b. [jun]$_1$[k]$_2$ [foo]$_3$[d]$_4$

Note that these are the pre-parenthesis and in-parenthesis text in (2a). The substrings surrounded by square brackets with the same number correspond to each other. Given such a correspondence, we can segment the pre-parenthesis text (3a) according to its English counterpart (3b), in which words

Table 4: Example of the substring alignment $\mathcal{A}$ between f ='ジャンクフード' and e ='junkfood' ($|\mathcal{A}| = 4$).

(f_i, e_i)	$\log p(f_i, e_i)$
(ジャン, jun)	-10.767
(ク, k)	-5.319
(フー, foo)	-11.755
(ド, d)	-5.178

are separated by white space. We can recognize that the katakana string 'ジャンク', which is the concatenation of the first two substrings in (3a), forms a single word because it corresponds to the English word *junk*, and so on. Consequently, (3a) can be segmented into two words, 'ジャンク (junk)' and 'フード (food)'. The word alignment is trivially established.

For problems (a) and (b), we can also use the phonetic similarity between pre-parenthesis and in-parenthesis text. If the parenthetical expression does not provide the transliteration, or if the left boundary is erroneously identified, we can expect the phonetic similarity to become small. Such situations thus can be identified.

The remainder of this section details this approach. Section 5.2 presents a probabilistic model for discovering substring alignment such as (3). Section 5.3 shows how to extract word-aligned transliteration pairs by using the probabilistic model.

5.2 Phonetic similarity model

To establish the substring alignment between katakana and Latin alphabet strings, we use the probabilistic model proposed by (Jiampojamarn et al., 2007). Let f and e be katakana and alphabet strings, and $\mathcal{A}$ be the substring alignment between them. More precisely, $\mathcal{A}$ is a set of corresponding substring pairs (f_i, e_i) such that $f = f_1 f_2 \dots f_{|\mathcal{A}|}$ and $e = e_1 e_2 \dots e_{|\mathcal{A}|}$. The probability of such alignment is defined as

$$\log p(f, e, \mathcal{A}) = \sum_{(f_i, e_i) \in \mathcal{A}} \log p(f_i, e_i).$$

Since $\mathcal{A}$ is usually unobservable, it is treated as a hidden variable. Table 4 illustrates an example of the substring alignment between f ='ジャンクフード' and e ='junkfood', and the likelihood of each substring pair estimated in our experiment.

The model parameters are estimated from a set of transliteration pairs (f, e) using the EM algorithm. In the E-step, we estimate $p(\mathcal{A}|f, e)$ based on the current parameters. In the parameter estimation, we restrict both f_i and e_i to be at most three characters long. Doing this not only makes the E-step computationally efficient but avoids over-fitting by forbidding too-long substrings to be aligned. In the M-step, the parameter is re-estimated using the result of the E-step. We can accomplish this by using an extension of the forward-backward algorithm. See (Jiampojamarn et al., 2007) for details.

Given a new transliteration pair (f, e), we can determine the substring alignment as

$$\mathcal{A}^* = \operatorname*{argmax}_{\mathcal{A}} \log p(f, e, \mathcal{A}).$$

In finding the substring alignment, a white space on the English side is used as a constraint, so that the English substring e_i does not span a white space.

5.3 Extracting word-aligned transliteration pairs

The word-aligned transliteration pairs are extracted using the phonetic similarity model, as follows.

First, candidate transliteration pairs (f, e) are extracted from the parenthetical expressions. This is done by extracting English words inside parentheses and pre-parenthesis text written in katakana. English words are normalized by lower-casing capital letters.

Second, we determine the left boundary by using the confidence score: $\frac{1}{N} \log p(f, e, \mathcal{A}^*)$, where N is the number of English words. The term $\frac{1}{N}$ prevents the score from being unreasonably small when there are many words. We truncate f by removing the leftmost characters one by one, until the confidence score exceeds a predefined threshold θ. If f becomes empty, the pair is regarded as a non-transliteration and discarded.

Finally, for the remaining pairs, the Japanese side is segmented and the word alignment is established according to $\mathcal{A}^*$. This results in a list of word-aligned transliteration pairs (Table 3).

6 Experiments and Discussion

We conducted experiments to investigate how the use of the paraphrasing and the back-transliteration

improves the performance of the discriminative model.

6.1 Experimental setting

To train the phonetic similarity model, we used a set of transliteration pairs extracted from the Wikipedia.[4] Since person names are almost always transliterated when they are imported from English into Japanese, we made use of the Wikipedia articles that belong to the *Living people* category. From the titles of those articles, we automatically extracted person names written in katakana, together with their English counterparts obtainable via the multilingual links provided by the Wikipedia. This yielded 17,509 transliteration pairs for training. In performing the EM algorithm, we tried ten different initial parameters and selected the model that achieved the highest likelihood.

The data for training and testing the perceptron was built using a Japanese-English dictionary EDICT.[5] We randomly extracted 5286 entries written in katakana from EDICT and manually annotated word boundaries by establishing word correspondences to their English transliterations. Since English transliterations are already provided by EDICT, the annotation can be trivially done by native speakers of Japanese. Using this data set, we performed 2-fold cross-validation for testing the perceptron. The number of iterations was set to 20 in all the experiments.

To compute the dictionary-based feature DICT(y) in our basic feature set, we used NAIST-jdic.[6] It is the largest dictionary used for Japanese word segmentation, and it includes 19,885 katakana words.

As Web corpora, we used 1.7 G sentences of blog articles. From the corpora, we extracted 14,966,205 (potential) paraphrases of katakana noun compounds together with their frequencies. We also extracted 151,195 word-aligned transliteration pairs. In doing this, we ranged the threshold θ in $\{-10, -20, \cdots -150\}$ and chose the value that performed the best ($\theta = -80$).

The results were evaluated using precision, recall, F_1-score, and accuracy. Precision is the number of correctly identified words divided by the number of all identified words, recall is the number of correctly identified words divided by the number of all oracle words, the F_1-score is their harmonic mean, and accuracy is the number of correctly split katakana noun compounds divided by the number of all the katakana noun compounds.

[4] http://ja.wikipedia.org/

[5] http://www.csse.monash.edu.au/~jwb/edict_doc.html

[6] http://sourceforge.jp/projects/naist-jdic

6.2 Baseline systems

We compared our system with three frequency-based baseline system, two supervised baselines, and two state-of-the-art word segmentation baselines. The first frequency-based baseline, UNIGRAM, performs compound splitting based on a word 1-gram language model (Schiller, 2005; Alfonseca et al., 2008b):

$$\boldsymbol{y}^* = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}(x)} \prod_i p(y_i),$$

where $p(y_i)$ represents the probability of y_i. The second frequency-based baseline, GMF, outputs the splitting option with the highest geometric mean frequency of the constituent words (Koehn and Knight, 2003):

$$\boldsymbol{y}^* = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}(x)} \mathrm{GMF}(\boldsymbol{y}) = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}(x)} \Big\{ \prod_i f(y_i) \Big\}^{1/|\boldsymbol{y}|},$$

where $f(y_i)$ represents the frequency of y_i. The third frequency-based baseline, GMF2, is a modification of GMF proposed by Nakazawa et al. (2005). It is based on the following score instead of GMF($\boldsymbol{y}$):

$$\mathrm{GMF2}(\boldsymbol{y}) = \begin{cases} \mathrm{GMF}(\boldsymbol{y}) & (|\boldsymbol{y}| = 1) \\ \frac{\mathrm{GMF}(\boldsymbol{y})}{\frac{C}{N^l} + \alpha} & (|\boldsymbol{y}| \geq 2), \end{cases}$$

where C, N, and α are hyperparameters and l is the average length of the constituent words. Following (Nakazawa et al., 2005), the hyperparameters were set as C = 2500, N = 4, and α = 0.7. We estimated $p(y)$ and $f(y)$ from the Web corpora.

The first supervised baseline, AP, is the averaged perceptron model trained using only the basic feature set. The second supervised baseline, AP+GMF2 is a combination of AP and GMF2, which performed the best amongst the frequency-based baselines. Following (Alfonseca et al.,

Table 5: Comparison with baseline systems.

Type	System	P	R	F_1	Acc
Frequency	UNIGRAM	64.2	49.7	56.0	63.0
	GMF	42.9	62.0	50.7	47.5
	GMF2	67.4	76.0	71.5	72.5
Supervised	AP	81.9	82.5	82.2	83.4
	AP+GMF2	83.0	83.9	83.4	84.2
	PROPOSED	**86.4**	**87.4**	**87.1**	**87.6**
Word seg.	JUMAN	71.4	60.1	65.3	69.8
	MECAB	72.4	73.7	67.8	71.6

2008b), GMF2 is integrated into AP as two binary features indicating whether GMF2($\boldsymbol{y}$) is larger than any other candidates, and whether GMF2($\boldsymbol{y}$) is larger than the non-split candidate. Although Alfonseca et al. (2008b) also proposed using (the log of) the geometric mean frequency as a feature, doing so degraded performance in our experiment.

Regarding the two state-of-the-art word segmentation systems, one is JUMAN,[7] a rule-based word segmentation system (Kurohashi and Nagao, 1994), and the other is MECAB,[8] a supervised word segmentation system based on CRFs (Kudo et al., 2004). These two baselines were chosen in order to show how well existing word segmentation systems perform this task. Although the literature states that it is hard for existing systems to deal with katakana noun compounds (Nakazawa et al., 2005), no empirical data on this issue has been presented until now.

6.3 Splitting result

Table 5 compares the performance of our system (PROPOSED) with the baseline systems. First of all, we can see that PROPOSED clearly improved the performance of AP, demonstrating the effectiveness of using paraphrases and back-transliterations.

Our system also outperformed all the frequency-based baselines (UNIGRAM, GMF, and GMF2). This is not surprising, since the simple supervised baseline, AP, already outperformed the unsupervised frequency-based ones. Indeed similar experimental results were also reported by Alfonseca (2008a). An interesting observation here is the comparison between PROPOSED and AP+GMF2. It reveals that our approach improved the performance of AP more than the frequency-based method did. These results indicate that paraphrasing and back-transliteration are more informative clues than the simple frequency of constituent words. We would like to note that the higher accuracy of PROPOSED in comparison with the baselines is statistically significant ($p < 0.01$, McNemar's test).

The performance of the two word segmentation baselines (JUMAN and MECAB) is significantly worse in our task than in the standard word segmentation task, where nearly 99% precision and recall are reported (Kudo et al., 2004). This demonstrates that splitting a katakana noun compound is not at all a trivial task to resolve, even for the state-of-the-art word segmentation systems. On the other hand, PROPOSED outperformed both JUMAN and MECAB in this task, meaning that our technique can successfully complement the weaknesses of the existing word segmentation systems.

By analyzing the errors, we interestingly found that some of the erroneous splitting results are still acceptable to humans. For example, while 'アップロード (upload)' was annotated as a single word in the test data, our system split it into 'アップ (up)' and 'ロード (load)'. Although the latter splitting may be useful in some applications, it is judged as wrong in our evaluation framework. This implies the importance of evaluating the splitting results in some extrinsic tasks. We leave it to a future work.

6.4 Investigation on out-of-vocabulary words

In our test data, 2681 out of the 5286 katakana noun compounds contained at least one out-of-vocabulary word that are not registered in NAIST-jdic. Table 6 illustrates the results of the supervised systems for those 2681 and the remaining 2605 katakana noun compounds (referred to as w/ OOV and w/o OOV data, respectively). While the accuracy exceeds 90% for w/o OOV data, it is substantially degraded for w/ OOV data. This is consistent with our claim that out-of-vocabulary words are a major source of errors in splitting noun compounds.

The three supervised systems performed almost equally for w/o OOV data. This is because AP trivially performs very well on this subset, and it is difficult to get any further improvement. On the other hand, we can see that there are substantial performance gaps between the systems for w/ OOV data. This result reflects the effect of the additional fea-

[7] http://nlp.kuee.kyoto-u.ac.jp/nl-resource/juman.html

[8] http://sourceforge.net/projects/mecab

Table 6: Splitting results of the supervised systems for w/ OOV and w/o OOV data.

System	w/ OOV data				w/o OOV data			
	P	**R**	**F_1**	**Acc**	**P**	**R**	**F_1**	**Acc**
AP	66.9	69.9	68.3	72.8	**95.4**	93.2	94.3	94.2
AP+GMF2	69.7	73.7	71.6	75.2	95.2	92.4	93.7	93.6
PROPOSED	**76.8**	**79.3**	**78.0**	**80.9**	95.3	**94.2**	**94.8**	**94.5**

tures more directly than is shown in table 5.

6.5 Effect of the two new features

To see the effect of the new features in more detail, we looked at the performances of our system using different feature sets (Table 7). The first column represents the feature set we used: BASIC, PARA, TRANS, and ALL represent the basic features, the paraphrase feature, the back-transliteration feature, and all the features. The results demonstrate that adding either of the new features improved the performance, and the best result was when they were used together. In all cases, the improvement over BASIC was statistically significant ($p < 0.01$, McNemar's test).

Next, we investigated the coverage of the features. Our test data comprised 7709 constituent words, 4937 (64.0%) of which were covered by NAIST-jdic. The coverage was significantly improved when using the back-transliteration feature. We observed that 6216 words (80.6%) are in NAIST-jdic or word-aligned transliteration pairs extracted from the Web text. This shows that the back-transliteration feature successfully reduced the number of out-of-vocabulary words. On the other hand, we observed that the paraphrase and back-transliteration features were activated for 79.5% (1926/2423) and 15.5% (376/2423) of the word boundaries in our test data.

Overall, we see that the coverage of these features is reasonably good, although there is still room for further improvement. It would be beneficial to use larger Web corpora or more paraphrase rules, for example, by having a system that automatically learns rules from the corpora (Barzilay and McKeown, 2001; Bannard and Callison-Burch, 2005).

6.6 Sensitivity on the threshold θ

Finally we investigated the influence of the threshold θ (Figure 1 and 2). Figure 1 illustrates the system performance in terms of F_1-score for different values of θ. While the F_1-score drops when the value of θ is too large (e.g., -20), the F_1-score is otherwise almost constant. This demonstrates it is generally easy to set θ near the optimal value. More importantly, the F_1-score is consistently higher than BASIC irrespective of the value of θ. Figure 2 represents the number of distinct word-aligned transliteration pairs that were extracted from the Web corpora. We see that most of the extracted transliteration pairs have high confidence score.

Table 7: Effectiveness of paraphrase (PARA) and back-transliteration feature (TRANS).

Feature set	P	R	F_1	Acc
BASIC	81.9	82.5	82.2	83.4
BASIC+PARA	85.1	85.3	85.2	85.9
BASIC+TRANS	85.1	86.3	85.7	86.5
ALL	**86.4**	**87.4**	**87.1**	**87.6**

7 Conclusion

In this paper, we explored the idea of using monolingual and bilingual paraphrases for splitting katakana noun compounds in Japanese. The experiments demonstrated that our method significantly improves the splitting accuracy by a large margin in comparison with the previously proposed methods. This means that paraphrasing provides a simple and effective way of using unlabeled textual data for identifying implicit word boundaries within katakana noun compounds.

Although our investigation was restricted to katakana noun compounds, one might expect that a similar approach would be useful for splitting other types of noun compounds (e.g., German noun compounds), or for identifying general word boundaries, not limited to those between nouns, in Asian languages. We think these are research directions worth exploring in the future.

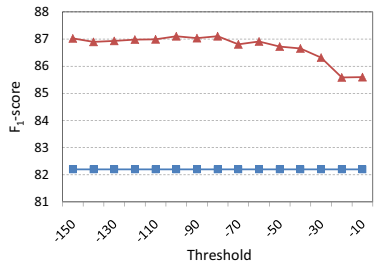

Figure 1: Influence of the threshold θ (x-axis) on the F_1-score (y-axis). The triangles and squares represent systems using the ALL and BASIC feature sets, respectively.

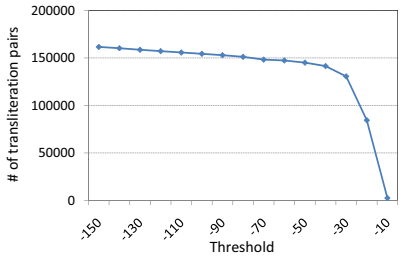

Figure 2: The number of distinct word-aligned transliterations pairs that were extracted from the Web corpora for different values of θ.

Acknowledgement

This work was supported by the *Multimedia Web Analysis Framework towards Development of Social Analysis Software* program of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

Enrique Alfonseca, Slaven Bilac, and Stefan Pharies. 2008a. Decompoundig query keywords from compounding languages. In *Proceedings of ACL, Short Papers*, pages 253–256.

Enrique Alfonseca, Slaven Bilac, and Stefan Pharies. 2008b. German decompounding in a difficult corpus. In *Proceedings of CICLing*, pages 128–139.

Rie Kubota Ando and Lillian Lee. 2003. Mostly-unsupervised statistical segmentation of Japanese Kanji sequences. *Natural Language Engineering*, 9(2):127–149.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of ACL*, pages 597–604.

Mohit Bansal and Dan Klein. 2011. Web-scale features for full-scale parsing. In *Proceedings of ACL*, pages 693–702.

Regina Barzilay and Kathleen McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of ACL*, pages 50–57.

Martin Braschler and Bärbel Ripplinger. 2004. How effective is stemming and decompounding for German text retrieval? *Information Retrieval*, 7:291–316.

Jamese Breen. 2009. Identification of neologisms in Japanese by corpus analysis. In *Proceedings of eLexicography in the 21st centry conference*, pages 13–22.

Eric Brill, Gray Kacmarcik, and Chris Brockett. 2001. Automatically harvesting katakana-English term pairs from search engine query logs. In *Proceedings of NLPRS*, pages 393–399.

Ralf D. Brown. 2002. Corpus-driven splitting of compound words. In *Proceedings of TMI*.

Guihong Cao, Jianfeng Gao, and Jian-Yun Nie. 2007. A system to mine large-scale bilingual dictionaries from monolingual Web pages. In *Proceedings of MT Summit*, pages 57–64.

Chris Dyer. 2009. Using a maximum entropy model to build segmentation lattices for MT. In *Proceedings of NAACL*, pages 406–414.

Yoav Freund and Robert E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

Florian Holz and Chris Biemann. 2008. Unsupervised and knowledge-free learning of compound splits and periphrases. In *CICLing*, pages 117–127.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignment and hidden Markov models to letter-to-phoneme conversion. In *HLT-NAACL*, pages 372–379.

Kyo Kageura, Fuyuki Yoshikane, and Takayuki Nozawa. 2004. Parallel bilingual paraphrase rules for noun compounds: Concepts and rules for exploring Web language resources. In *Proceedings of Workshop on Asian Language Resources*, pages 54–61.

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

Philip Koehn and Kevin Knight. 2003. Empirical methods for compound splitting. In *Proceedings of EACL*, pages 187–193.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of EMNLP*, pages 230–237.

Sadao Kurohashi and Makoto Nagao. 1994. Improvements of Japanese morphological analyzer JUMAN. In *Proceedings of the International Workshop on Sharable Natural Language Resources*, pages 22–38.

Toshiaki Nakazawa, Daisuke Kawahara, and Sadao Kurohashi. 2005. Automatic acquisition of basic Katakana lexicon from a given corpus. In *Proceedings of IJCNLP*, pages 682–693.

Preslav Nakov and Marti Hearst. 2005a. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *Proceedings of CoNLL*, pages 17–24.

Preslav Nakov and Marti Hearst. 2005b. Using the Web as an implicit training set: Application to structural ambiguity resolution. In *Proceedings of HLT/EMNLP*, pages 835–342.

Jong-Hoon Oh and Hitoshi Isahara. 2008. Hypothesis selection in machine transliteration: A Web mining approach. In *Proceedings of IJCNLP*, pages 233–240.

Naoaki Okazaki, Sophia Ananiadou, and Jin'ichi Tsujii. 2008. A discriminative alignment model for abbreviation recognition. In *Proceedings of COLING*, pages 657–664.

Anne Schiller. 2005. German compound analysis with wfsc. In *Proceedings of Finite State Methods and Natural Language Processing*, pages 239–246.

Ariel S. Schwartz and Marti A. Hearst. 2003. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Proceedings of PSB*, pages 451–462.

Natsuko Tsujimura. 2006. *An Introduction to Japanese Linguistics*. Wiley-Blackwell.

Xianchao Wu, Naoaki Okazaki, and Jun'ichi Tsujii. 2009. Semi-supervised lexicon mining from parenthetical expressions in monolingual Web pages. In *Proceedings of NAACL*, pages 424–432.