Keyboard Logs as Natural Annotations for Word Segmentation

Fumihiko Takahashi*
Yahoo Japan Corporation
Midtown Tower, 9-7-1 Akasaka, Minato-ku,
Tokyo, Japan
ftakahas@yahoo-corp.jp

Shinsuke Mori
ACCMS, Kyoto University
Yoshida Honmachi, Sakyo-ku,
Kyoto, Japan
forest@i.kyoto-u.ac.jp

Abstract

In this paper we propose a framework to improve word segmentation accuracy using input method logs. An input method is software used to type sentences in languages which have far more characters than the number of keys on a keyboard. The main contributions of this paper are: 1) an input method server that proposes word candidates which are not included in the vocabulary, 2) a publicly usable input method that logs user behavior (like typing and selection of word candidates), and 3) a method for improving word segmentation by using these logs. We conducted word segmentation experiments on tweets from Twitter, and showed that our method improves accuracy in this domain. Our method itself is domain-independent and only needs logs from the target domain.

1 Introduction

The first step of almost all natural language processing (NLP) for languages with ambiguous word boundaries (such as Japanese and Chinese) is solving the problem of word identification ambiguity. This task is called word segmentation (WS) and the accuracy of state-of-the-art methods based on machine learning techniques is more than 98% for Japanese and 95% for Chinese (Neubig et al., 2011; Yang and Vozila, 2014). Compared to languages like English with clear word boundaries, this ambiguity poses an additional problem for NLP tasks in these languages. To make matters worse, the domains of the available training data often differ from domains where there is a high demand for NLP, which causes a severe degradation in WS performance. Examples include machine translation of patents, text mining of medical texts, and marketing on the micro-blog site Twitter[1]. Some papers have reported low accuracy on WS or the joint task of WS and part-of-speech (POS) tagging of Japanese or Chinese in these domains (Mori and Neubig, 2014; Kaji and Kitsuregawa, 2014; Liu et al., 2014)

To cope with this problem, we propose a way to collect information from people as they type Japanese or Chinese on computers. These languages use far more characters than the number of keys on a keyboard, so users use software called an input method (IM) to type text in these languages. Unlike written texts in these languages, which lack word boundary information, text entered with an IM can provide word boundary information that can used by NLP systems. As we show in this paper, logs collected from IMs are a valuable source of word boundary information.

An IM consists of a client (front-end) and a server (back-end). The client receives a key sequence typed by the user and sends a phoneme sequence (*kana* in Japanese or *pinyin* in Chinese) or some predefined commands to the server. The server converts the phoneme sequence into normal written text as a word sequence or proposes word candidates for the phoneme sequence in the region specified by the user. We noticed that the actions performed by people using the IM (such as typing and selecting word candidates) provide information about word boundaries, including context information.

In this paper, we first describe an IM for Japanese which allows us to collect this information. We then propose an automatic word segmenter that uses IM logs as a language resource to improve its performance. Finally, we report experimental results showing that our method increases the accuracy of a word segmenter on Twitter texts by using logs collected from a browser add-on ver-

*This work was done when the first author was at Kyoto University.

[1]https://twitter.com/ (accessed in 2015 May).

Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1186–1196,
Lisbon, Portugal, 17-21 September 2015.

sion of our IM.

The three main contributions of this paper are:

- an IM server that proposes word candidates which are not included in the vocabulary (Section 3),
- a publicly usable IM that logs user behavior (such as typing and selection of word candidates) (Section 4),
- a method for improving word segmentation by using these logs (Section 5).

To the best of our knowledge, this is the first paper proposing a method for using IM logs to successfully improve WS.

2 Related Work

The main focus of this paper is WS. Corpus-based, or empirical, methods were proposed in the early 90's (Nagata, 1994). Then (Mori and Kurata, 2005) extended it by lexicalizing the states like many researches in that era, grouping the word-POS pairs into clusters inspired by the class-based n-gram model (Brown et al., 1992), and making the history length variable like a POS tagger in English (Ron et al., 1996). In parallel, there were attempts at solving Chinese WS in a similar way (Sproat and Chang, 1996). WS or the joint task of WS and POS tagging can be seen as a sequence labeling problem. So conditional random fields (CRFs) (Peng et al., 2004; Lafferty et al., 2001) have been applied to this task and showed better performance than POS-based Markov models (Kudo et al., 2004). The training time of sequence-based methods tends to be long, especially when we use partially annotated data. Thus a simple method based on pointwise classification has been shown to be as accurate as sequence-based methods and fast enough to make active learning practically possible (Neubig et al., 2011). Since the pointwise method decides whether there is a word boundary or not between two characters without referring to other word boundary decisions in the same sentence, it is straightforward to train the model from partially annotated sentences. We adopt this WS system for our experiments.

Along with the evolution of models, the NLP community has become increasingly aware of the importance of language resources (Neubig and Mori, 2010; Mori and Neubig, 2014). So Mori and Oda (2009) proposed to incorpolate dictionaries for human into a WS system with a different word definition. CRFs were also extended to enable training from partially annotated sentences (Tsuboi et al., 2008). When using partially annotated sentences for WS training data, word boundary information exists only between some character pairs and is absent for others. This extension was adopted in Chinese WS to make use of so-called natural annotations (Yang and Vozila, 2014; Jiang et al., 2013). In that work, tags in hyper-texts were regarded as annotations and used to improve WS performance. The IM logs used in this paper are also classified as natural annotations, but contain much more noise. In addition, we need an IM that is specifically designed to collect logs as natural annotations.

Server design is the most important factor in capturing information from IM logs. The most popular IM servers are based on statistical language modeling (Mori et al., 1999; Chen and Lee, 2000; Maeta and Mori, 2012). Their parameters are trained from manually segmented sentences whose words are annotated with phoneme sequences, and from sentences automatically annotated with NLP tools which are also based on machine learning models trained on the annotated sentences. Thus normal IM servers are not capable of presenting out-of-vocabulary (OOV) words (which provide large amounts of information on word boundaries) as conversion candidates. To make our IM server capable of presenting OOV words, we extend a statistical IM server based on (Mori et al., 2006), and ensure that it is computationally efficient enough for practical use by the public.

The target domain in our experiments is Twitter, a site where users post short messages called tweets. Since tweets are an immediate and powerful reflection of public attitudes and social trends, there have been numerous attempts at extracting information from them. Examples include information analysis of disasters (Sakai et al., 2010), estimation of depressive tendencies (Tsugawa et al., 2013), speech diarization (Higashinaka et al., 2011), and many others. These works require preprocessing of tweets with NLP tools, and WS is the first step. So it is clear that there is strong demand for improving WS accuracy. Another reason why we have chosen Twitter for the test domain is that the tweets typed using our server are open and

we can avoid privacy problems. Our method does not utilize any other characteristics of tweets. So it also works in other domains such as blogs.

3 Input Method Suggesting OOV Words

In this section we propose a practical statistical IM server that suggests OOV word candidates in addition to words in its vocabulary.

3.1 Statistical Input Method

An input method (IM) is software which converts a phoneme sequence into a word sequence. This is useful for languages which contain far more characters than keys on a keyboard. Since there are some ambiguities in conversion, a conversion engine based on a word n-gram model has been proposed (Chen and Lee, 2000). Today, almost all IM engines are based on statistical methods.

For the LM unit, instead of words we propose to adopt word-pronunciation pairs $u = \langle \boldsymbol{y}, w \rangle$. Thus given a phoneme sequence $\boldsymbol{y}_1^l = y_1 y_2 \cdots y_l$ as the input, the goal of our IM engine is to output a word sequence $\hat{\boldsymbol{w}}_1^m$ that maximizes the probability $P(\boldsymbol{w}, \boldsymbol{y}_1^l)$ as follows:

$$\begin{aligned} \hat{\boldsymbol{w}}_1^m &= \operatorname*{\mathbf{argmax}}_{\boldsymbol{w}} P(\boldsymbol{w}, \boldsymbol{y}_1^l), \\ P(\boldsymbol{w}, \boldsymbol{y}_1^l) &= \prod_{i=1}^{m+1} P(u_i | \boldsymbol{u}_{i-n+1}^{i-1}), \end{aligned}$$

where the concatenation of $\boldsymbol{y}_i$ in each u_i is equal to the input: $\boldsymbol{y}_1^l = \boldsymbol{y}_1 \boldsymbol{y}_2 \cdots \boldsymbol{y}_m$. In addition u_j ($j \leq 0$) are special symbols introduced to simplify the notation and u_{m+1} is a special symbol indicating a sentence boundary.

As in existing statistical IM engines, parameters are estimated from a corpus whose sentences are segmented into words annotated with their pronunciations as follows:

$$P(u_i | \boldsymbol{u}_{i-n+1}^{i-1}) = \frac{F(\boldsymbol{u}_{i-n+1}^{i})}{F(\boldsymbol{u}_{i-n+1}^{i-1})}, \qquad (1)$$

where $F(\cdot)$ denotes the frequency of a pair sequence in the corpus. In contrast to IM engines based on a word n-gram model, ours does not require an additional model describing relationships between words and pronunciations, and thus it is much simpler and more practical.

Existing statistical IM engines only need an accurate automatic word segmenter to estimate the parameters of the word n-gram model. As the equation above shows, our pair-based engine also needs an accurate way of automatically estimating pronunciation (phoneme sequences). However, recently an automatic pronunciation estimator (Mori and Neubig, 2011) that delivers as accurate as state-of-the-art word segmenters has been proposed. As we explain in Section 6, in our experiments both our IM engine and existing ones delivered accuracy of 91%.

3.2 Enumerating Substrings as Candidate Words

Essentially, the IM engine which we have explained above does not have the ability to enumerate words which are unknown to the word segmenter and the pronunciation estimator used to build the training data. The aim of our research is to gather language information from user behavior as they use an IM. So we extend the basic IM engine to enumerate all the substrings in a corpus with all possible pronunciations. For that purpose, we adopt the notion of a stochastically segmented corpus (SSC) (Mori and Takuma, 2004) and extend it to the pronunciation annotation to words.

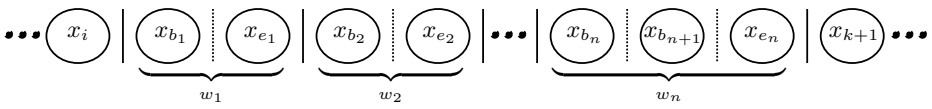
3.2.1 Stochastically Segmented Corpora

An SSC is defined as a combination of a raw corpus C_r (hereafter referred to as the character sequence $\boldsymbol{x}_1^{n_r}$) and word boundary probabilities of the form P_i, which is the probability that a word boundary exists between two characters x_i and x_{i+1}. These probabilities are estimated by a model based on logistic regression (LR)(Fan et al., 2008) trained on a manually segmented corpus referring to the same features as those used in (Neubig et al., 2011). Since there are word boundaries before the first character and after the last character of the corpus, $P_0 = P_{n_r} = 1$. Then word n-gram frequencies on an SSC are calculated as follows:

Word 0-gram frequency: This is defined as the expected number of words in the SSC:

$$f(\cdot) = 1 + \sum_{i=1}^{n_r - 1} P_i.$$

Word n-gram frequency ($n \geq 1$): Consider the situation in which a word sequence $\boldsymbol{w}_1^n$ occurs in the SSC as a subsequence beginning at the $(i+1)$-th character and ending at the k-th character and each word w_m in the word sequence is equal to the character sequence beginning at the



$$f_r(w_1^n) = P_i(1-P_{b_1})P_{e_1}(1-P_{b_2})P_{e_2}\cdots(1-P_{b_n})(1-P_{b_{n+1}})P_{e_n}$$

Figure 1: Word n-gram frequency in a stochastically segmented corpus.

b_m-th character and ending at the e_m-th character ($\boldsymbol{x}_{b_m}^{e_m} = w_m,\ 1 \leq \forall m \leq n;\ e_m + 1 = b_{m+1},\ 1 \leq \forall m \leq n-1;\ b_1 = i+1;\ e_n = k$) (See Figure 1 for an example). The word n-gram frequency of a word sequence $f_r(\boldsymbol{w}_1^n)$ in the SSC is defined by the summation of the stochastic frequency at each occurrence of the character sequence of the word sequence $\boldsymbol{w}_1^n$ over all of the occurrences in the SSC:

$$f_r(\boldsymbol{w}_1^n) = \sum_{(i,\boldsymbol{e}_1^n)\in O_n} P_i \left[\prod_{m=1}^{n} \left\{ \prod_{j=b_m}^{e_m-1} (1-P_j) \right\} P_{e_m} \right],$$

where $\boldsymbol{e}_1^n = (e_1, e_2, \cdots, e_n)$ and $O_n = \{(i, e_1^n) | \boldsymbol{x}_{b_m}^{e_m} = w_m, 1 \leq m \leq n\}$.

We calculate word n-gram probabilities by dividing word n-gram frequencies by word $(n-1)$-gram frequencies. For a detailed explanation and a mathematical proof of this method, please refer to (Mori and Takuma, 2004).

3.2.2 Pseudo-Stochastically Segmented Corpora

The computational costs (in terms of both time and space) for calculating an n-gram model from an SSC are very high[2], so it is not a practical technique for implementing an IM engine. In order to reduce the computational costs we approximate an SSC using a deterministically tagged corpus, which is called a pseudo-stochastically segmented corpus (pSSC) (Kameko et al., 2015). The following is the method for producing a pSSC from an SSC.

- For i = 1 to $n_r - 1$
  1. output a character x_i,
  2. generate a random number $0 \leq p < 1$,
  3. output a word boundary if $p < P_i$ or output nothing otherwise.

Now we have a corpus in the same format as a standard segmented corpus with variable (non-constant) segmentation.

[2]This is because an SSC has many words and word fragments. Additionally, word n-gram frequencies must be calculated using floating point numbers instead of integers.

3.2.3 Pseudo-Stochastically Tagged Corpora

We can annotate a word with its all possible pronunciations and their probabilities, as is done in an SSC. We call a corpus containing sequences of words ($w_1 w_2 \cdots w_i \cdots$) annotated with a sequence of pairs of a pronunciation and its probability ($\langle \boldsymbol{y}_{i,1}, p_{i,1} \rangle, \langle \boldsymbol{y}_{i,2}, p_{i,2} \rangle, \cdots$, where $\sum_j p_{i,j} = 1$, for $\forall i$) a stochastically tagged corpus (STC)[3]. We can estimate these probabilities using an LR model built from sentences annotated with pronunciations (Mori and Neubig, 2011).

Similar to pSSC we then produce a pseudo-stochastically tagged sentence (pSTC) from an STC as follows:

- For each w_i in the sentence
  1. generate a random number $0 \leq p < 1$,
  2. annotate w_i with its j-th phoneme sequence $\boldsymbol{y}_{i,j}$, where $\sum_1^{j-1} p_{i,j} \leq p < \sum_1^{j} p_{i,j}$

Now we have a corpus in the same format as a standard corpus annotated with variable pronunciation.

By estimating the parameters in Equation (1) from a pSTC derived from a pSSC, our IM engine can also suggest OOV word candidates with various possible segmentation and pronunciations without incurring high computational costs.

3.2.4 Suggestion of OOV Words

Here we give an intuitive explanation why our IM engine can suggest OOV words for a certain phoneme sequence. Let us take an OOV word example: "横アリ/yo-ko-a-ri," an abbreviation of "横浜アリーナ" (Yokohama city arena). A WS system tends to segment it into "横" (side) and "アリ" (ant) because they are frequent nouns. In a pSSC, however, some occurrences of the string "横アリ" are remain concatenated as the correct word. For pronunciation, the first character has two possible pronunciations "yo-ko" and "o-u."

[3]Because the existence or non-existence of a word boundary information can also be expressed as a tag, a stochastically tagged corpus includes stochastic segmentation.

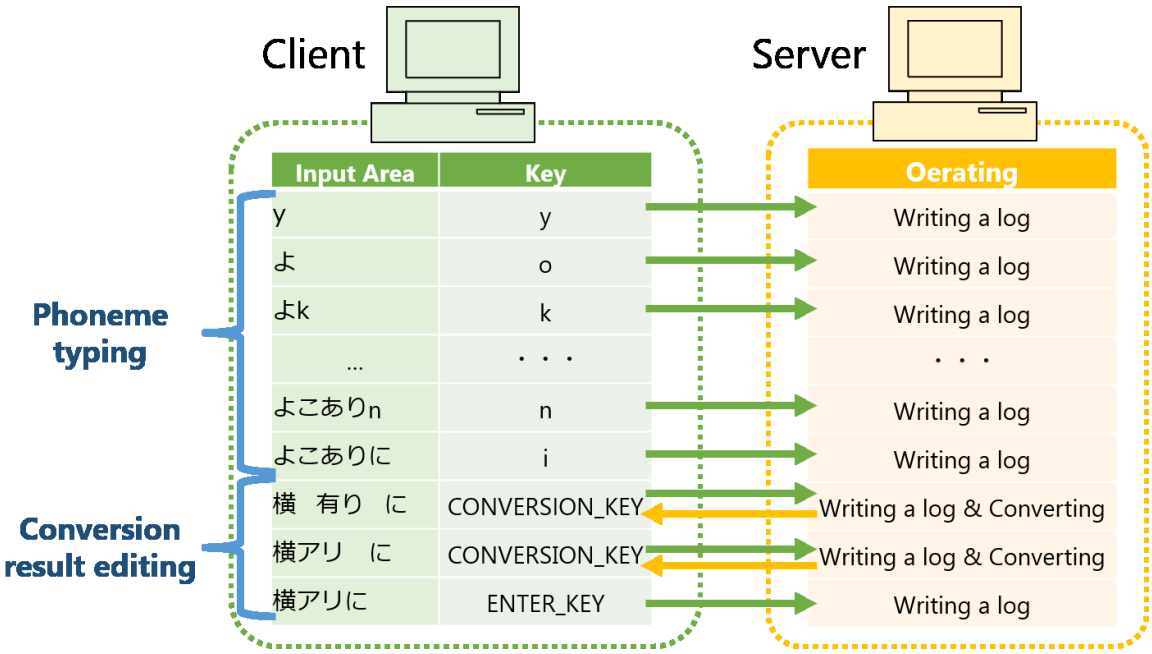


Figure 2: Input method for collecting logs.

So deterministic pronunciation estimation of this new word has the risk of outputting the erroneous result "o-u-a-ri." This prevents our engine from presenting "横アリ" as a conversion candidate for the input "yo-ko-a-ri." The pSTC, however, contains two possible pronunciations for this word and allows our engine to present the OOV word "横アリ" for the input "yo-ko-a-ri."

Thus when the user of our IM engine types "yo-ko-a-ri-ni-i-ku" and selects "横アリ に (to) 行く (go)," the engine can learn an OOV word "横アリ/yo-ko-a-ri" with context "に/ni 行く/i-ku".

4 Input Method Logs

In this section we first propose an IM which allows us to collect user logs. We then examine the characteristics of these logs and some difficulties in using them as language resources.

4.1 Collecting Logs from an Input Method

As Figure 2 shows, the client of our IM, running on the user's PC, is used to input characters and to modify conversion results. The server logs both input from the client and the results of conversions performed in response to requests from the client.

Our IM has two phases: phoneme typing and conversion result editing. In each phase, the client sends the typed keys to the server with a timestamp and its IP address.

Phoneme typing: First the user inputs ASCII characters for a phoneme sequence. If the phoneme sequence itself is what the user wants to write, the user may not go to the next phase. The server records the keys typed to enter the phoneme sequence, cursor movements, and the phoneme sequence if the user selects it as-is.

Conversion result editing: Then the user presses a space key to make the IM engine convert the phoneme sequence to the most likely word sequence based on Equation (1). Sometimes the user changes some word boundaries, makes the IM engine enumerate candidate words covering the region, and selects the intended one from the list of candidates. The server records a space key and the final word sequence.

4.2 Characteristics of Input Method Logs

Table 1 shows an example of interesting log messages from the same IP address[4]. In many cases, users type sentence fragments but not a complete sentence. So in the example there are six fragments within a short period indicated by the timestamps. If the user selects the phoneme sequence as-is without going to the conversion result editing phase, we can expect that there are word boundaries on both sides of the phoneme sequence. In-

[4]In reality, logs from different IPs are stored in the order that they were received.

Table 1: Input method logs of a tweet '横アリに比べると安めかと' (It is cheap compared with Yokohama arena).

Timestamp	Phoneme sequence	Edit result	Note
18:37:11.21	よこありに/yo-ko-a-ri-ni	横アリ/yo-ko-a-ri に/ni	(with Yokohama arena)
18:37:12.60	くらっべる/ku-ra-b-be-ru	くらっ/ku-ra-b ベル/be-ru	Mistyping
18:37:14.94	くらべる/ku-ra-be-ru	比べ/ku-ra-be る/ru	Revised input (compare)
18:37:15.32	と/to	N/A	(*inflectional ending*)
18:37:19.82	ものの/mo-no-no	N/A	Discarded in the twitter
18:37:22.42	やすめかと/ya-su-me-ka-to	安め/ya-su-me か/ka と/to	(cheap)

side the phoneme sequence, however, there is no information. If the user goes to the conversion result editing phase, we can expect that the final word sequence has correct word boundary information.

There are two main problems that make it difficult to directly use IM logs as a training corpus for word segmentation. The first problem is fragmentation. IM users send the phoneme sequences for sentence fragments to the engine to avoid editing long conversion results that require many cursor movements. Thus the phoneme sequence and the final word sequence tend to be sentence fragments (as we noted above) and as a result they lose context information. The second problem is noise. Word boundary information is unreliable even when it is present because of mistakenly selected conversions or words entered separately. From these observations, the IM logs are treated as partially segmented sentence fragments that include noise.

5 Word Segmentation Using Input Method Logs

In this section we first explain various ways to generate language resources for a word segmenter from IM logs. We then describe an automatic word segmenter which utilizes these resources. In the examples below we use the three-valued notation (Mori and Oda, 2009) to denote partial segmentation as follows:

- | : there is a word boundary,
- – : there is not a word boundary,
- ␣ : there is no information.

5.1 Input Method Logs as Language Resources

The phoneme sequences and edit results in the final selection themselves are considered to be partially segmented sentences. We call the corpus generated directly from the logs "**Log-as-is**." Examples in Table 1 are converted as following.

Example of Log-as-is (12 annotations)

横–ア–リ | に
く–ら–っ | べ–ル
比–べ | る
と
も␣の␣の
安–め | か | と

Here the number of annotations is the sum of "–" and "|". In this example, one entry corresponds to one entry of the training data for the word segmenter. As you can easily imagine, Log-as-is may contain mistaken results (noise) and short entries (fragmentation). Both are harmful for a word segmenter.

To cope with the fragmentation problem, we propose to connect some logs based on their timestamps. If the difference between the timestamps of two sequential logs is short, both logs are probably from the same sentence. So we connect two sequential logs if the time difference between the last key of the first log and the first key of the second log is smaller than a certain threshold s. In the experiment we set $s = 500[\mathtt{ms}]$ based on observations of our behavior[5]. This method is referred to as "**Log-chunk**." Using this method, we obtain the following from the examples in Table 1.

Example of Log-chunk (15 annotations)

横–ア–リ | に | く–ら–っ | べ–ル
比–べ | る | と | も␣の␣の
安–め | か | と

We see that Log-chunk contains more context information than Log-as-is.

For preventing the noise problem, we propose to filter out logs with a small number of conversions. We expect that an edited sentence will have many OOV words and not much noise. Therefore we use logs which were converted more than n_c times. In the experiment we set $n_c = 2$ based on

[5]The results were stable for s in preliminary experiments.

Table 2: Corpus specifications.

	#sent.	#words	#char.
Training			
BCCWJ	56,753	1,324,951	1,911,660
Newspaper	8,164	240,097	361,843
Conversation	11,700	147,809	197,941
Test			
BCCWJ-test	6,025	148,929	212,261
TWI-test	2,976	37,010	58,316

observations of our behavior[6]. This method is referred to as "**Log-mconv**." Using this method, the examples in Table 1 becomes the following.

Example of Log-mconv (3 annotations)
横–ア–リ | に

As this example shows, Log-mconv contains short entries (fragmentation) like Log-as-is. However, we expect that the annotated tweets do not include mistaken boundaries or conversions that were discarded.

Obviously we can combine Log-chunk and Log-mconv to avoid both the fragmentation and noise problems. This combination is referred to as "**Log-chunk-mconv**."

5.2 Training a Word Segmenter on Logs

The IM logs give us partially segmented sentence fragments, so we need a word segmenter capable of learning from them. We can use a word segmenter based on a sequence classifier (Tsuboi et al., 2008; Yang and Vozila, 2014; Jiang et al., 2013) or one based on a pointwise classifier (Neubig et al., 2011). Although both types are viable, we adopt the latter in the experiments because it requires much less training time while delivering comparable accuracy.

Here is a brief explanation of the word segmenter based on the pointwise method. For more detail the reader may refer to (Neubig et al., 2011). The input is an unsegmented character sequence $\boldsymbol{x} = x_1 x_2 \cdots x_k$. The word segmenter decides if there is a word boundary $t_i = 1$ or not $t_i = 0$ by using support vector machines (SVMs) (Fan et al., 2008)[7]. The features are character n-grams and character type n-grams ($n = 1, 2, 3$) around the decision points in a window with a width of 6 characters. Additional features are triggered if character n-grams in the window match with character sequences in the dictionary. This approach is called pointwise because the word boundary decision is made without referring to the other decisions on the points $j \neq i$. As you can see from the explanation given above, we can also use partially segmented sentences from IM logs for training in the standard way.

Table 3: Language resources derived from logs.

	#sentence fragments	#annotations
Log-as-is	32,119	39,708
Log-chunk	8,685	63,144
Log-mconv	4,610	10,852
Log-chunk-mconv	1,218	14,242

6 Evaluation

As an evaluation of our methods, we measured the accuracy of WS without using logs (the baseline) and using logs converted by several methods. There are two test corpora: one is the general domain corpus from which we built the baseline WS, and the other is the same domain that the IM logs were collected from, Twitter.

6.1 Corpora

The annotated corpus we used to build the baseline word segmenter is the manually annotated part (core data) of the Balanced Corpus of Contemporary Written Japanese (BCCWJ) (Maekawa, 2008), plus newspaper articles and daily conversation sentences. We also used a 234,652-word dictionary (UniDic) provided with the BCCWJ. A small portion of the BCCWJ core data is reserved for testing. In addition, we manually segmented sentences randomly obtained from Twitter[8] during the same period as the log collection for the test corpus. Table 2 shows the details of these corpora.

[6]The results were stable for n_c in the preliminary experiments.

[7]The reason why we use SVM for word segmentation is that the accuracy is generally higher than that based on LR. It was so in the experiments of this paper. The F-measure of LR on TWI-test was 91.30 (Recall = 89.50, Precision = 93.17), which is lower than that of SVM (see Table 4). To make an SSC, however, we use an LR model because we need word boundary probabilities.

[8]We extracted body text from 1,592 tweets excluding mentions, hash tags, URLs, and ticker symbols. Then we divided the body text into sentences by separating on newline characters, resulting in 2,976 sentences.

Table 4: WS accuracy on the tweets.

	Recall [%]	Precision [%]	F-measure
Baseline	90.31	94.05	92.14
+ Log-as-is	90.33	93.77	92.02
+ Log-chunk	91.04	94.29	92.64
+ Log-mconv	90.62	94.09	92.32
+ Log-chunk-mconv	91.40	94.45	92.90

Table 5: WS accuracy on BCCWJ.

	Recall [%]	Precision [%]	F-measure
Baseline	99.01	98.97	98.99
+ Log-as-is	99.02	98.87	98.94
+ Log-chunk	99.05	98.88	98.96
+ Log-mconv	98.98	98.91	98.95
+ Log-chunk-mconv	98.98	98.92	98.95

6.2 Models using Input Method Logs

To make the training data for our IM server, we first chose randomly selected tweets (786,331 sentences) in addition to the unannotated part of the BCCWJ (358,078 sentences). We then trained LR models which estimate word boundary probabilities and pronunciation probabilities for words (and word candidates) from the training data shown in Table 2 and UniDic. We made a pSTC for our IM engine from 1,207,182 sentences randomly obtained from Twitter by following the procedure which we explained in Subsection 3.2.3[9].

We launched our IM as a browser add-on for Twitter and collected 19,770 IM logs from 7 users between April 24 and December 31, 2014. Following the procedures in Section 5.1, we obtained the language resources shown in Table 3. We combined them with the training corpus and dictionaries to build four WSs, which we compared with the baseline.

6.3 Results and Discussion

Following the standard in WS experiments, the evaluation criteria are recall, precision, and F-measure (their harmonic mean). Recall is the number of correctly segmented words divided by the number of words in the test corpus. Precision is the number of correctly segmented words divided by the number of words in the system output.

Table 4 and 5 show WS accuracy on TWI-test and BCCWJ-test, respectively. The difference in accuracy of the baseline method on BCCWJ-test and TWI-test shows that WS of tweets is very difficult. The fact that the precision on TWI-test is much higher than the recall indicates that the baseline model suffers from over-segmentation. This over-segmentation problem is mainly caused by OOV words being divided into known words. For example, “横アリ” (Yokohama arena) is divided into the two know words “横” (side) and “アリ” (ant).

When we compare the F-measures on TWI-test, all the models referring to the IM logs outperform the baseline model trained only from the BCCWJ. The highest is the Log-chunk-mconv model and the improvement over the baseline is statistically significant (significance level: 1%). In addition the accuracies of the five methods on the BCCWJ (Table 5) are almost the same and there is no statistical significance (significance level: 1%) between any two of them.

We analyzed the words misrecognized by the WSs, which we call error words. Table 6 shows the number of error words, the number of OOV words, and the ratio of OOV words to error words. Here the vocabulary is the set of the words appearing in the training data or in UniDic (see Table 2). Although the result of the WS trained on Log-as-is contains more error words than the baseline, the OOV ratio is less than the baseline. This means that the IM logs have a potential to reduce errors caused by OOV words.

Table 6 also indicates that the best method Log-chunk-mconv had the greatest success in reducing

[9]There is no overlap with the test data.

Table 6: Ratio of OOV words in error words.

	#Error words	#OOV words	(ratio[%])
Baseline	446	103	(23.09)
+ Log-as-is	467	89	(19.06)
+ Log-chunk	428	81	(18.93)
+ Log-mconv	443	88	(19.86)
+ Log-chunk-mconv	413	74	(17.79)

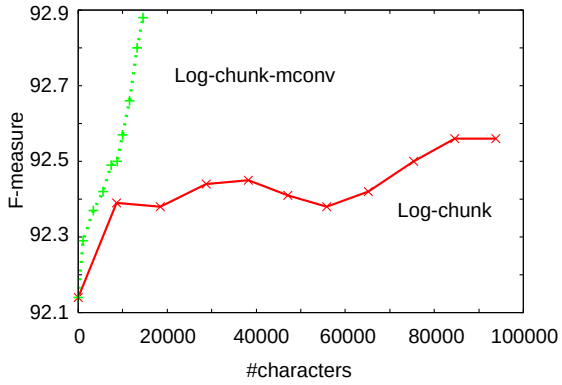


Figure 3: Relationship between WS accuracy on the tweets and log size.

errors caused by OOV words. However, the majority of error words are in-vocabulary words. It can be said that our log chunking method (Log-chunk or Log-chunk-mconv) enabled the WSs to eliminate many known word errors by using context information.

To investigate the impact of the log size, we measured WS accuracy on TWI-test when varying the log size during training. Figure 3 shows the results. Table 4 says that Log-chunk-mconv and Log-chunk increase the accuracy nicely. The graph, however, clarifies that Log-chunk-mconv achieves high accuracy with fewer training data converted from logs. In other words, the method Log-chunk-mconv is good at distilling the informative parts and filtering out the noisy parts. These characteristics are very important properties to have as we consider deploying our IM to a wider audience. An IM is needed to type Japanese and the number of Japanese speakers is more than 100 million. If we can use input logs of even 1% of them for the same or longer period[10], the idea we propose in this paper can improve WS accuracy on various domains efficiently and automatically.

As a final remark, this paper describes a successful example of how to build a useful tool for the NLP community. This process has three steps: 1) design a useful NLP application that can collect user logs, 2) deploy it for public use, and 3) devise a method for mining data from the logs.

[10]The number of users using our system in this paper is 7 for 8 months.

7 Conclusion

This paper described the design of a publicly usable IM which collects natural annotations for use as training data for another system. Specifically, we (1) described how to construct an IM server that suggests OOV word candidates, (2) designed a publicly usable IM that collects logs of user behavior, and (3) proposed a method for using this data to improve word segmenters. Tweets from Twitter are a promising source of data with great potential for NLP, which is one reason why we used them as the target domain for our experiments. The experimental results showed that our methods improve accuracy in this domain. Our method itself is domain-independent and only needs logs from the target domain, so it is worth testing on other domains and with much longer periods of data collection.

Acknowledgments

This work was supported by JSPS Grants-in-Aid for Scientific Research Grant Number 26280084 and Microsoft CORE project. We thank Dr. Hisami Suzuki, Dr. Koichiro Yoshino, and Mr. Daniel Flannery for their valuable comments and suggestions on the manuscript. We are also grateful to the anonymous users of our input method.

References

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Zheng Chen and Kai-Fu Lee. 2000. A new statistical approach to Chinese pinyin input. In *Proceedings*

of the 38th Annual Meeting of the Association for Computational Linguistics, pages 241–247.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Ryuichiro Higashinaka, Noriaki Kawamae, Kugatsu Sadamitsu, Yasuhiro Minami, Toyomi Meguro, Kohji Dohsaka, and Hirohito Inagaki. 2011. Building a conversational model from two-tweets. *IEEE Transactions on ASRU*, pages 330–335.

Wenbin Jiang, Meng Sun, Yajuan Lu, Yating Yang, and Qun Liu. 2013. Discriminative learning with natural annotations: Word segmentation as a case study. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 761–769.

Nobuhiro Kaji and Masaru Kitsuregawa. 2014. Accurate word segmentation and POS tagging for Japanese microblogs: Corpus annotation and joint modeling with lexical normalization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 99–109.

Hirotaka Kameko, Shinsuke Mori, and Yoshimasa Tsuruoka. 2015. Can symbol grounding improve low-level NLP? Word segmentation as a case study. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Applying conditional random fields to Japanese morphological analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 230–237.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth ICML*, pages 282–289.

Yijia Liu, Yue Zhang, Wangxiang Che, Ting Liu, and Fan Wu. 2014. Domain adaptation for CRF-based Chinese word segmentation using free annotations. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 864–874.

Kikuo Maekawa. 2008. Balanced corpus of contemporary written Japanese. In *Proceedings of the 6th Workshop on Asian Language Resources*, pages 101–102.

Hirokuni Maeta and Shinsuke Mori. 2012. Statistical input method based on a phrase class n-gram model. In *Workshop on Advances in Text Input Methods*.

Shinsuke Mori and Gakuto Kurata. 2005. Class-based variable memory length markov model. In *Proceedings of the InterSpeech2005*, pages 13–16.

Shinsuke Mori and Graham Neubig. 2011. A pointwise approach to pronunciation estimation for a TTS front-end. In *Proceedings of the InterSpeech2011*, pages 2181–2184.

Shinsuke Mori and Graham Neubig. 2014. Language resource addition: Dictionary or corpus? In *Proceedings of the Nineth International Conference on Language Resources and Evaluation*, pages 1631–1636.

Shinsuke Mori and Hiroki Oda. 2009. Automatic word segmentation using three types of dictionaries. In *Proceedings of the Eighth International Conference Pacific Association for Computational Linguistics*, pages 1–6.

Shinsuke Mori and Daisuke Takuma. 2004. Word n-gram probability estimation from a Japanese raw corpus. In *Proceedings of the Eighth International Conference on Speech and Language Processing*, pages 1037–1040.

Shinsuke Mori, Tsuchiya Masatoshi, Osamu Yamaji, and Makoto Nagao. 1999. Kana-kanji conversion by a stochastic model. *Transactions of Information Processing Society of Japan*, 40(7):2946–2953. (in Japanese).

Shinsuke Mori, Daisuke Takuma, and Gakuto Kurata. 2006. Phoneme-to-text transcription system with an infinite vocabulary. In *Proceedings of the 21st International Conference on Computational Linguistics*, pages 729–736.

Masaaki Nagata. 1994. A stochastic Japanese morphological analyzer using a forward-DP backward-A* n-best search algorithm. In *Proceedings of the 15th International Conference on Computational Linguistics*, pages 201–207.

Graham Neubig and Shinsuke Mori. 2010. Word-based partial annotation for efficient corpus construction. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation*, pages 2723–2727.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 529–533.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the 20th International Conference on Computational Linguistics*, pages 562–568.

Dana Ron, Yoram Singer, and Naftali Tishby. 1996. The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, 25:117–149.

Takeshi Sakai, Makoto Okazaki, and Yutaka Matsuo. 2010. Earthquake shakes Twitter users: Real-time event detection by social sensors. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, pages 851–860.

Richard Sproat and Chilin Shih William Gale Nancy Chang. 1996. A stochastic finite-state word-segmentation algorithm for Chinese. *Computational Linguistics*, 22(3):377–404.

Yuta Tsuboi, Hisashi Kashima, Shinsuke Mori, Hiroki Oda, and Yuji Matsumoto. 2008. Training conditional random fields using incomplete annotations. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 897–904.

Sho Tsugawa, Yukiko Mogi, Yusuke Kikuchi, Fumio Kishino, Kazuyuki Fujita, Yuichi Itoh, and Hiroyuki Ohsaki. 2013. On estimating depressive tendencies of Twitter users utilizing their tweet data. In *VR'13*, pages 1–4.

Fan Yang and Paul Vozila. 2014. Semi-supervised Chinese word segmentation using partial-label learning with conditional random fields. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 90–98.