Discriminative Method for Japanese Kana-Kanji Input Method

Hiroyuki Tokunaga **Daisuke Okanohara**
Preferred Infrastructure Inc.
2-40-1 Hongo, Bunkyo-ku, Tokyo
113-0033, Japan
{tkng,hillbig}@preferred.jp

Shinsuke Mori
Kyoto University
Yoshidahonmachi, Sakyo-ku, Kyoto
606-8501, Japan
forest@i.kyoto-u.ac.jp

Abstract

The most popular type of input method in Japan is *kana-kanji conversion*, conversion from a string of kana to a mixed kanji-kana string.

However there is no study using discriminative methods like structured SVMs for kana-kanji conversion. One of the reasons is that learning a discriminative model from a large data set is often intractable. However, due to progress of recent researches, large scale learning of discriminative models become feasible in these days.

In the present paper, we investigate whether discriminative methods such as structured SVMs can improve the accuracy of kana-kanji conversion. To the best of our knowledge, this is the first study comparing a generative model and a discriminative model for kana-kanji conversion. An experiment revealed that a discriminative method can improve the performance by approximately 3%.

1 Introduction

Kana-kanji conversion is conversion from kana characters to kanji characters, the most popular way of inputting Japanese text from keyboards. Generally one kana string corresponds to many kanji strings, proposing what the user wants to input is not trivial. We showed how input keys are processed in Figure 1.

Two types of problems are encountered in kana-kanji conversion. One is how to reduce conversion errors, and the other is how to correct smoothly when a conversion failure has occurred. In the present study, we focus on the problem of reducing conversion errors.

Early kana-kanji conversion systems employed heuristic rules, such as maximum-length-word matching.

With the growth of statistical natural language processing, data-driven methods were applied for kana-kanji conversion. Most existing studies on kana-kanji conversion have used probability models, especially generative models. Although generative models have some advantages, a number of studies on natural language tasks have shown that discriminative models tend to overcome generative models with respect to accuracy.

However, there have been no studies using only a discriminative model for kana-kanji conversion. One reason for this is that learning a discriminative model from a large data set is often intractable. However, due to progress in recent research on machine learning, large-scale learning of discriminative models has become feasible.

The present paper describes how to apply a discriminative model for kana-kanji conversion. The remainder of the present paper is organized as follows. In the next section, we present a brief description of Japanese text input and define the kana-kanji conversion problem. In Section 3, we describe related researches. Section 4 provides the baseline method of the present study, i.e., kana-kanji conversion based on a probabilistic language model. In Section 5, we describe how to apply the discriminative method for kana-kanji conversion. Section 6 presents the experimental results, and conclusions are presented in Section 7.

2 Japanese Text Input and Kana-Kanji Conversion

Japanese text is composed of several scripts. The primary components are hiragana, katakana, (we refer them as *kana*) and kanji.

The number of kana is less than hundred. In many case, we input romanized kana characters from the keyboard. One of the task of a Japanese

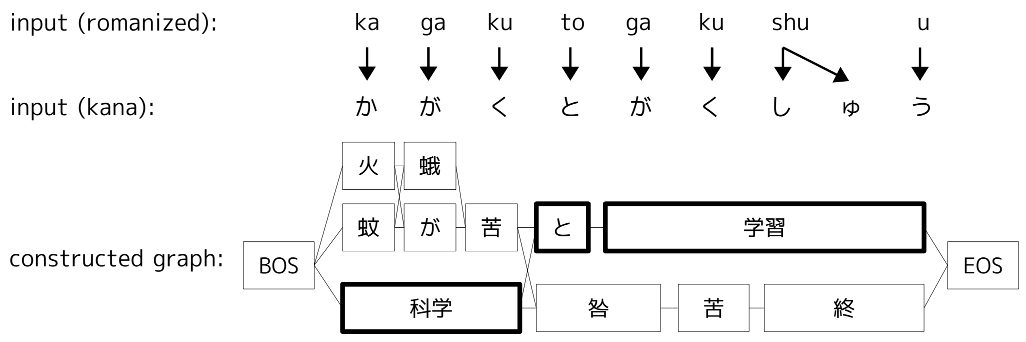

Figure 1: A graph constructed from an input string. Some nodes are omitted for simplicity.

input method is transliterating them to kana.

The problem is how to input kanji. The number of kanji is more than ten thousand, the ordinal keyboards in Japan do not have a sufficient number of keys to enable the user to input such characters directly.

In order to address this problem, a number of methods have been proposed and investigated. Handwriting recognition systems, for example, have been successfully implemented. Another successful method is kana-kanji conversion, which is currently the most commonly used method for inputting Japanese text due to its fast input speed and low initial cost to learn.

2.1 Kana-kanji conversion

Kana-kanji conversion is the problem of converting a given kana string into a mixed kanji-kana string. For simplicity, in the following we describe a mixed kanji-kana string as *a kanji string*.

What should be noted here is that kana-kanji conversion is the conversion from a kana *sentence* to a kanji *sentence*. One of the key points of kana-kanji conversion is that an entire sentence can be converted at once. This is why kana-kanji conversion is great at input speed.

Since an input string is non-segmented sentence, every kana-kanji conversion system must be able to segment a kana sentence into words. This is not a trivial problem, recent kana-kanji conversion softwares often employ the statistical methods.

Although there is a measure of freedom with respect to the design of a kana-kanji conversion system, in the present paper, we discuss kana-kanji conversion systems they comply with the following procedures:

1. Construct a graph from the input string.

   We must first construct a graph that represents all possible conversions. An example of such a graph is given in Figure 1.

   We must use a dictionary in order to construct this graph.

   Since all edges are directed from the start node to the goad node, the graph is a directed acyclic graph (DAG).

2. Select the most likely path in the graph.

   This task is broken into two parts. The first is setting the cost of each vertex and edge. The cost is often determined by supervised learning methods. The second is finding the most likely path from the graph. Viterbi algorithm is used for this task.

Formally, the task of kana-kanji conversion can be defined as follows. Let $\boldsymbol{x}$ be an input, an unsegmented sentence written in kana. Let $\boldsymbol{y}$ be a path, i.e., a sequence of words. When we write the jth word in $\boldsymbol{y}$ as y_j, $\boldsymbol{y}$ can be denoted as $\boldsymbol{y} = y_1 y_2 \ldots y_n$, where n is the number of words in path $\boldsymbol{y}$.

Let $\mathcal{Y}$ be a set of candidate paths in the DAG built from the input sentence $\boldsymbol{x}$. The goal is to select a correct path $\boldsymbol{y}^*$ from all candidate paths in $\mathcal{Y}$.

2.2 Parameter estimation with a probabilistic language model

If we defined the kana-kanji conversion as a graph search problem, all edges and vertices must have costs. There are two ways to estimate these parameters. One is to use a language model, which was proposed in the 1990s, and the other is to use a discriminative model, as in the present study.

This subsection describes kana-kanji conversion based on probabilistic language models, treated as a baseline in the present paper.

In this method, given a kana string, we calculate the probabilities for each conversion candidate and then present the candidate that has the highest probability.

We denote the probability P of a conversion candidate $\boldsymbol{y}$ given $\boldsymbol{x}$ by $P(\boldsymbol{y}|\boldsymbol{x})$.

Using Bayes' theorem, we can transform this expression as follows:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|\boldsymbol{y})P(\boldsymbol{y})}{P(\boldsymbol{x})} \propto P(\boldsymbol{y})P(\boldsymbol{x}|\boldsymbol{y}),$$

where $P(\boldsymbol{y})$ is the *language model*, and $P(\boldsymbol{x}|\boldsymbol{y})$ is the *kana-kanji model*.

The language model $P(\boldsymbol{y})$ assigns a high probability to word sequences that are likely to occur. Word n-gram models or class n-gram models are often used in the natural language processing.

The definition of n-gram model is denoted as follows:

$$P(\boldsymbol{y}) = \prod_j P(y_j|y_{j-1}, y_{j-2}, \ldots, y_{j-n+1}),$$

where $n-1$ indicates the length of the history used to estimate the probability.

If we adopt a word bigram ($n = 2$) model, then $P(\boldsymbol{y})$ is decomposed into the following form:

$$P(\boldsymbol{y}) = \prod_j P(y_j|y_{j-1}), \tag{1}$$

where $P(y_j|y_{j-1})$ is the probability of the appearance of y_j after y_{j-1}. Let $c(y)$ be the number of occurrences of y in the corpus, and let $c(y_1, y_2)$ be the number of occurrences of y_1 after y_2 in the corpus. The probability $P(y_j|y_{j-1})$ is calculated as follows:

$$P(y_j|y_{j-1}) = \frac{c(y_j, y_{j-1})}{c(y_{j-1})}.$$

The kana-kanji model $P(\boldsymbol{x}|\boldsymbol{y})$ is assigned a high probability if $\boldsymbol{x}$ corresponds to $\boldsymbol{y}$ several times. In Japanese, most characters have multiple pronunciations. For example, 雨 (rain) could be read as "ame", "same" or "u". In the case of 雨, most of Japanese expect the reading to be "ame". Therefore P(ame| 雨) should be assigned a higher probability than for "same" or "u".

Mori et al. (1999) proposed the following decomposition model for kana-kanji model:

$$P(\boldsymbol{x}|\boldsymbol{y}) = \prod_j P(x_j|y_j)$$

where each $P(x_j|y_j)$ is a fraction of how many times the word y_j is read as x_j. Given that $d(x_j, y_j)$ is the number of times word y_j is read as x_j, $P(x_j|y_j)$ can be written as follows:

$$P(x_j|y_j) = \frac{d(x_j, y_j)}{\sum_i d(x_i, y_j)}.$$

2.3 Smoothing

In Eq. (1), if any $P(y_j|y_{j-1})$ is zero, $P(\boldsymbol{y})$ is also estimated to zero. This means that $P(\boldsymbol{y})$ is always zero if a word that does not appear in the training corpus is appeared in $\boldsymbol{y}$. Since we use a language model to determine which sentence is more natural as a Japanese sentence, both probabilities being zero does not help us. This is referred to as the *zero frequency problem*.

We apply *smoothing* to prevent this problem. In the present paper, we implemented both of *additive smoothing* and *interpolated Kneser-Ney smoothing* (Chen and Goodman, 1998; Teh, 2006). Surprisingly, the additive smoothing overcame the interpolated Kneser-Ney smoothing. Therefore we adopt the additive smoothing in the experiments.

3 Related Research

Mori et al. (1999) proposed a kana-kanji conversion method based on a probabilistic language model. They used the class bigram as their language model.

Gao et al. (2006) reported the results of applying a discriminative language model for kana-kanji conversion. Their objective was domain adaptation using a discriminative language model for reranking of top-k conversion candidates enumerated by a generative model.

Kana-kanji conversion and morphological analysis are similar in some respects. Most notably, both are the same type of extension of the sequential labeling problem. Therefore, it would be worthwhile to consider studies on morphological analysis.

Nagata (1994) showed that a pos-trigram language model-based morphological analyzer achieved approximately 95% precision and recall, which was a state-of-the-art result at that time. Ten years later, Kudo el al. (2004) applied the conditional random field (CRF) to Japanese morphological analysis. They reported that this major discriminative probabilistic model does not suffer from label bias and length bias and is superior to the hidden Markov model (HMM) and maximum entropy Markov model (MEMM) with respect to accuracy.

The purpose of the present study is to investigate whether this increase in performance for morphological analysis also applies to kana-kanji conversion.

4 Kana-Kanji Conversion Based on Discriminative Methods

In this section, we present a description of kana-kanji conversion based on discriminative methods.

In discriminative methods, we calculate a score for each conversion candidate $\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3 \ldots$ for input $\boldsymbol{x}$. The candidate that has the highest score is presented to the user.

We herein restrict the score function such that the score function can be decomposed into weighted sum of K feature functions $\boldsymbol{\Psi}$, where $\boldsymbol{\Psi}$ is a vector of each feature function Ψ_k. We also restrict arguments of feature functions to $\boldsymbol{x}, y_{j-1}, y_j$ in order to use the Viterbi algorithm for fast conversion. A feature function $\Psi_k(\boldsymbol{x}, y_{j-1}, y_j)$ returns 1 if the feature is enabled, otherwise $\Psi_k(\boldsymbol{x}, y_{j-1}, y_j)$ returns 0.

The score of a conversion candidate $\boldsymbol{y}$ over $\boldsymbol{x}$ is calculated as follows:

$$f(\boldsymbol{x}, \boldsymbol{y}) = \sum_j \sum_k w_k \Psi_k(\boldsymbol{x}, y_{j-1}, y_j),$$

where w_k is the weight of feature function Ψ_k, and $\boldsymbol{w}$ is a vector of each feature weight w_k.

Since the output of kana-kanji conversion is a vector, the problem is a structured output problem, which can be addressed in a number of ways, including the use of CRF (Lafferty et al., 2001) or SSVM (Tsochantaridis et al., 2005; Ratliff et al., 2007).

The performances of CRF, SSVM and other learner models are similar if all of the models use the same feature set (Keerthi and Sundararajan, 2007). We use the SSVM as the learner because it is somewhat easier to implement.

4.1 Structured SVM

The SSVM is a natural extension of the SVM for structured output problems.

We denote the ith datum as $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$. Here, $\mathcal{L}_i(\boldsymbol{y})$ is the loss for the ith datum, and we assume that $\mathcal{L}_i(\boldsymbol{y}) \geq 0$ for all $\boldsymbol{y} \neq \boldsymbol{y}^{(i)}$, and that $\mathcal{L}_i(\boldsymbol{y}^{(i)}) = 0$. Note that the value of $\mathcal{L}_i$ is zero if and only if $\boldsymbol{y} = \boldsymbol{y}^{(i)}$. This means that all of other $\boldsymbol{y}$ are treated as negative examples.

We adopt the following loss function, which is similar to Hamming loss:

$$\mathcal{L}_i(\boldsymbol{y}) = \sum_j l(y_j),$$

where $l(y_j)$ is 1 if the path is incorrect, otherwise $l(y_j)$ is 0.

The objective function of SSVM is expressed as follows:

$$\frac{1}{n} \sum_{i=1}^{n} r_i(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|, \tag{2}$$

where $r_i(\boldsymbol{w})$ is the risk function, which is defined as follows:

$$\max_{\boldsymbol{y} \in \mathcal{Y}} (\boldsymbol{w} f(\boldsymbol{x}^{(i)}, \boldsymbol{y}) + \mathcal{L}(\boldsymbol{y})) - \boldsymbol{w} f(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}). \tag{3}$$

Note that the *loss* and the *risk* are differentiated in the structured output problems, while they are often not in binary classification problems.

Here, $\|\boldsymbol{w}\|$, which is the norm of $\boldsymbol{w}$, is referred to as a regularization term. The most commonly used norms are L1-norm and L2-norm. We used L1-norm in the experiment because it tends to find sparse solutions. Since input methods are expected to work with modest amounts of RAM, this property is important. L1-norm is calculated as follows:

$$\|\boldsymbol{w}\|_1 = \sum_k |w_k|. \tag{4}$$

The positive real number λ is a parameter that trades off between the loss term and the regularization term.

In order to minimize this objective function, we used *FOBOS* as the parameter optimization method (Duchi and Singer, 2009).

4.2 Learning of the SSVM using FOBOS

FOBOS is a versatile optimization method for non-smooth convex problems, which can be used both online and batch-wise. We used online FOBOS for parameter optimization.

Algorithm 1 Learning of SSVM

```
for $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$ do
    $\boldsymbol{y}^* = \mathrm{argmax}_{\boldsymbol{y}} f(\boldsymbol{x}^{(i)}, \boldsymbol{y}) + \mathcal{L}(\boldsymbol{x}^{(i)}, \boldsymbol{y})$
    if $\boldsymbol{y}^* \neq \boldsymbol{y}^{(i)}$ then
        $\boldsymbol{w}^{i+\frac{1}{2}} = \boldsymbol{w}^{(i)} - \eta_t \nabla(f(\boldsymbol{x}^{(i)}, \boldsymbol{y}^*) - f(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}))$
        for $k \in K$ do
            $w_k^{(i+1)} = \mathrm{sign}(w_k^{i+\frac{1}{2}}) \max\{|w_k^{(i+\frac{1}{2})}| - \lambda\eta_t, 0\}$
        end for
    end if
end for
```

In the case of online FOBOS, FOBOS is viewed as an extension of the *stochastic gradient decent* and the *subgradient method*.

FOBOS alternates between two phases. The first step processes the (sub)gradient descent, the second step processes the regularization term in a manner similar to projected gradients.

The first step of FOBOS is as follows:

$$\boldsymbol{w}^{(i+\frac{1}{2})} = \boldsymbol{w}^{(i)} - \eta_i \boldsymbol{g}_i^f, \tag{5}$$

where η_t is the learning rate and $\boldsymbol{g}_t^f$ is a subgradient of the risk function.

The second step of FOBOS is defined as the following optimization problem:

$$\boldsymbol{w}^{(i+1)} = \underset{\boldsymbol{w}}{\mathrm{argmin}} \left\{ \frac{1}{2} \|\boldsymbol{w} - \boldsymbol{w}^{(i+\frac{1}{2})}\|^2 + \eta_{i+\frac{1}{2}} \|\boldsymbol{w}\| \right\}. \tag{6}$$

If the regularization term is L1-norm or L2-norm, the closed-form solutions are easily derived. For the case of the L1-norm, each element of vector $\boldsymbol{w}^{(i+1)}$ is calculated as follows:

$$w_k^{(i+1)} = \mathrm{sign}(w_k^{i+\frac{1}{2}}) \max\{|w_k^{(i+\frac{1}{2})}| - \lambda\eta_t, 0\}, \tag{7}$$

where sign is a function which returns 1 if the argument is greater than 0 and otherwise returns -1.

We present a pseudo code of SSVM by FOBOS as Algorithm 1.

In general, execution of the following expression needs exponential amount of calculation.

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\mathrm{argmax}} f(\boldsymbol{x}^{(i)}, \boldsymbol{y}) + \mathcal{L}(\boldsymbol{x}^{(i)}, \boldsymbol{y}). \tag{8}$$

Name	# sentences	Data Source
OC	6,476	Yahoo! Chiebukuro (Q&A site in Yahoo! Japan)
OW	5,934	White Book
PN	17,007	News Paper
PB	10,347	Book

Table 1: Details of Data Set

However, we restricted the form of our feature functions to $\Psi_k(\boldsymbol{x}, y_{j-1}, y_j)$ so that we can use the *Viterbi algorithm* to obtain $\boldsymbol{y}^*$. The time complexity of the Viterbi algorithm is linear, proportional to the length of $\boldsymbol{y}$.

Here, ∇f denotes a subgradient of function f. The derived subgradient for f is as follows:

$$\nabla f(\boldsymbol{x}, \boldsymbol{y}) = \sum_j \sum_k \Psi_k(\boldsymbol{x}, y_{j-1}, y_j).$$

Therefore, the first parameter update rule of FOBOS can be rewritten as follows:

$$\begin{aligned} \boldsymbol{w}^{i+\frac{1}{2}} =& \boldsymbol{w}^{(i)} - \eta_t(\sum_j \sum_k \Psi_k(\boldsymbol{x}, y_{j-1}^*, y_j^*) \\ & - \sum_j \sum_k \Psi_k(\boldsymbol{x}, y_{j-1}^{(i)}, y_j^{(i)})). \end{aligned} \tag{9}$$

5 Evaluation

In order to evaluate our method, we compared the generative model explained in Section 2 and our discriminative model explained in Section 4 using a popular data set.

5.1 Settings

As the data set, we used the balanced corpus of contemporary written Japanese (BCCWJ) (Maekawa, 2008). The corpus contains several different data set, and we used human-annotated data sets in our experiments. The human-annotated part of the BCCWJ consists of four parts, referred to as OC, OW, PN and PB. Each data set is summarized in Table 1.

In addition, we constructed a data set (referred to as ALL) that is the concatenation of OC, OW, PN and PB.

The baseline method we used herein is a language model based generative model. The language model is the linear sum of logarithm of a

Type	Template
Word Unigram	$\langle y_i \rangle$
Word Bigram	$\langle y_{i-1}, y_i \rangle$
Class Bigram	$\langle POS_{i-1}, POS_i \rangle$
Word and the Read	$\langle y_i, x_i \rangle$

Table 2: Feature Templates

class bigram probability, a word bigram probability and a word unigram probability. The smoothing method is additive smoothing, where $\delta = 10^{-5}$. The performance was insensitive to the δ when the value was small enough.

As the discriminative model, we used SSVM. The learning loop of the SSVM was repeated until convergence, i.e., 30 times for each data set. The learning rate η is a fixed float number, 0.1. We used L1-norm with $\lambda = 10^{-7}$ as the regularization term.

We implemented our SSVM learner in C language, the calculation time for the ALL data set was approximately 43 minutes and 20 seconds using an Intel Core 2 Duo (3.16GHz).

All of the experiments were carried out by five-fold cross validation, and each data set was randomly shuffled before being dividing into five data sets.

5.2 Feature functions

We summarized feature functions which we used in the experiments in Table 2. We used the second level part of speech (In some Japanese dictionaries, part of speech is designed to have hierarchical structure) as classes.

5.3 Criteria

We evaluated these methods based on precision, recall, and F-score, as calculated from the given answers and system outputs.

In order to compute the precision and recall, we must define true positive. In the present paper, we use the longest common subsequence of a given answer sentence and a system output sentence as true positive. Let N_{LCS} be the length of the longest common subsequence of a given answer and a system output. Let N_{DAT} be the length of the given answer sentence, and let N_{SYS} be the length of the system output sentence.

The definitions of precision, recall, and F-score are as follows:

$$\text{precision} = \frac{N_{LCS}}{N_{DAT}},$$
$$\text{recall} = \frac{N_{LCS}}{N_{SYS}},$$
$$\text{F-score} = 2\,\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

5.4 Difference between the discriminative model and the generative model

We compared the performance of the SSVM and the generative method based on a language model (baseline).

The results of the experiment were shown in Table 3. The precision, recall, and F-score for the SSVM and a baseline model are listed.

The experimental results revealed that the SSVM performed better than the baseline model for all data sets. However, the increase in performance for each data set was not uniform. The largest increase in performance was obtained for PN, whose data source is newspaper articles. Since newspaper articles are written by well-schooled newspersons, sentences are clear and consistent. Compared to newspapers, other data sets are noisy and inconsistent. The small improvement in performance is interpreted as being due to the data set being noisy and the relative difficulty in improving the performance scores as compared to newspapers.

5.5 Relationship between data set size and performance

In order to investigate the effect on performance change related to data set size, we examined the performance of the SSVM and the baseline model for each data set size. The ALL data set is used in this experiment.

The results are shown in Figure 2. The horizontal axis denotes the number of sentences used for training, and the vertical axis denotes the F-score.

The SSVM consistently outperforms the baseline model whereas the number of sentences is more than roughly 1000.

Interestingly, the baseline model performed better than the SSVM, where the data set size is relatively small. Generative models are said to be superior to discriminative models if there is only a small amount of data (Ng and Jordan, 2002). The result of the present experiment agrees naturally with their observations.

	baseline						SSVM					
	Precision		Recall		F-score		Precision		Recall		F-score	
	avg.	SD	avg.	SD	avg.	SD	avg.	SD	avg.	SD	avg.	SD
OC	87.4	0.31	86.9	0.17	87.2	0.22	**88.0**	0.41	**89.1**	0.27	**88.5**	0.33
OW	93.7	0.09	93.1	0.12	93.4	0.10	**96.1**	0.09	**96.4**	0.13	**96.2**	0.10
PN	87.4	0.11	86.4	0.16	86.9	0.13	**91.1**	0.24	**91.6**	0.17	**91.4**	0.20
PB	87.8	0.13	86.9	0.15	87.3	0.13	**89.5**	0.24	**90.3**	0.28	**89.9**	0.24
ALL	88.6	0.08	87.2	0.12	87.9	0.10	**92.2**	0.16	**92.4**	0.21	**92.3**	0.19

Table 3: Performance comparison for the SSVM and the baseline with the language model. (SD: standard deviation.) Performance is measured by precision, recall and F-score.

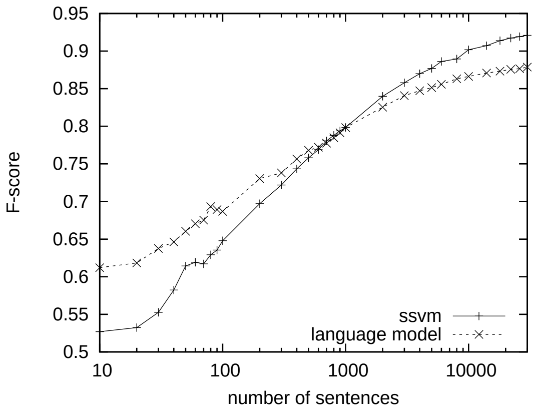

Figure 2: F-score vs. data set size. With the increase of the data set size, SSVM has overcome baseline method.

5.6 Examples of misconversions

In this subsection, we present examples of misconversions, which are categorized into four types.

Mori et al. (1999) categorized misconversions into these three types, and their categorization was also applicable to the proposed system. In addition, we present a number of misconversions that could not be categorized into three categories.

5.6.1 Homonym failures

Japanese has numerous homonyms. For correct conversion, syntactically and semantically correct homonyms must be chosen.

Corpus 多分 衛星 放送 で やっ て いる の だ と 思い ます 。
Perhaps that show is broadcast by satellite.

System 多分 衛生 放送 で やっ て いる の だ と 思い ます 。
Perhaps that show is broadcast by health.

The system failed to recognize the compound word *satellite broadcast*. This type of errors will decrease as the data set size increases.

Corpus 暴力 団 の 抗争 も 激化 。
Bloody conflicts of gang is also escalated.

System 暴力 団 の 構想 も 激化 。
Concept of gang is also escalated.

Since '暴力 団' (gang) and '抗争' (bloody conflicts) are strongly correlated, this problem would be solved if we can use a feature which considers long-distance information.

In principle, some fraction of homonym failures could not be solved because of ambiguity of kana string. A typical example is the name of a person. The number of conversion candidates of 'Hiroyuki' is over 100.

5.6.2 Unknown word failures

It is difficult to convert a word which is not in the dictionary. This type of misconversions cannot be reduced with the SSVM of the present study. In the following, we present an example of unknown word failure.

Corpus 家 で チキン ナゲット 作れ ます か ？
Can you make chicken nuggets at home?

System 家 で チキン な ゲット 作れ ます か ？
Can you make chicken ??? at home?

The underlined part of system output is broken as Japanese, and it cannot be converted into English. This type of errors often causes misdetections of word boundaries. This error is caused by the absence of particles ナゲット from the dictionary.

5.6.3 Orthographic variant failures

Some words have only one meaning and one pronunciation, but have multiple expressions. Numbers are examples of such words. As a typical case, six could be denoted as *six* or *6* or *VI*. In addition, in Japanese, six can also be denoted as '六' (roku) or '陸' (roku).

Some of these expression are misconversions. For example, '陸' is seldom used, and in most cases converting 'roku' as '陸' would be considered a misconversion. Nevertheless, with the exception of human judgment, we have no way to distinguish misconversions from non-misconversions. Thus, in the present paper, all orthographic variants are treated as failures.

Corpus 一 歳年下の弟は中学三年になる ところ だった.

System 1 歳年下の弟は中学三年になる 所 だった.

5.6.4 Other failures

Failures that do not fit into any of the above three categories are salient in these experiments. The following are two examples:

Corpus 太 すぎ ず 、 細 すぎ ない ジーンズ 。
Not too thick, not too thin jeans.

System 太 すぎ 図 、 細 すぎ ない ジーンズ 。
Too thick figure, not too thin jeans.

The reason of misconversion is that the score for '図' is too high.

Corpus ようや く 来 た か って 感じ です 。
I feel that it has finally come.

System よう や 茎 たかっ て 感じ です 。

The score for '茎' (kuki/caulome) is too high. In this case, misconversion is accompanied by serious word boundary detection errors, and most of the system output is difficult to interpret.

These errors are caused by poorly estimated parameters.

5.6.5 Discussion of misconversions

Although the discriminative method could improves the performance of kana-kanji conversion if there is sufficient data, there are still misconversions.

Based on the investigation of the misconversions, if a much larger data set is used, several misconversions will be vanish. In fact, there are several errors that do not exist in the closed test results.

However, there are some types of errors that can not be eliminated just by using a large amount of data. Some of these errors will vanish if we can use long-distance information in the feature functions.

6 Conclusion

In the present paper, we suggested the possibility of the discriminative methods for kana-kanji conversion.

We proposed a system using a SSVM with FOBOS for parameter optimization. The experiments of the present study revealed that the discriminative method is 1 to 4% superior with respect to precision and recall.

One of the advantages of the discriminative methods is the flexibility allowing the inclusion of a variety of feature functions. However, we used only the set of a kana, a word surface and a class (part of speech) in the experiments. using the entire input string is expected to reduce homonym failures, and further exploration of this area would be interesting.

The data set used in the present study was modest size. The increase in performance due to a large data set should be investigated in the future. In general, a large annotated data set is difficult to obtain. There are numbers of ways to tackle the problem. There are two important options, one is applying of semi-supervised learning, another is use of a morphological analyzer. We will choose the latter option because building an affective semi-supervised discriminative learning model would be difficult for the case of kana-kanji conversion.

Since the optimization method used in the present study is online learning, the optimization method can also be used for personalization from the correction operations log of the user. There have been few studies on this subject, and there has been no report of discriminative models being used. Learning from the correction log of the user is difficult because users often make mistakes. Kana-kanji conversion software users sometimes complain about the degradation of the conversion performance as a side effect of personalization. Therefore, an error detection mechanism will be important. In the future, we plan to implement a complete Japanese input method that embeds the kana-kanji conversion system developed for the present paper. Moreover, we intend to take into account statistics and investigate input errors.

References

Stanley F. Chen and Joshua T. Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report Technical Report TR-10-98, Computer Science Group, Harvard University.

John Duchi and Yoram Singer. 2009. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934.

Jianfeng Gao, Hisami Suzuki, and Bin Yu. 2006. Approximation lasso methods for language modeling. In *Annual Meeting of the Association for Computational Linguistics*, Sydney, Australia, July.

Selvaraj Sathiya Keerthi and Sellamanickam Sundararajan. 2007. Crf versus svm-struct for sequence labeling. *Yahoo Research Technical Report*.

Taku Kudo, Kaoru Yamamoto, and Yuji Matsumoto. 2004. Appliying conditional random fields to japanese morphological analysis. In *Conference on Empirical Methods in Natural Language Processing*, Barcelona, Spain, July.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*.

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

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

Masaaki Nagata. 1994. A stochastic japanese morphological analyzer using a forward-dp backward-a* n-best search algorithm. pages 201–207, 8.

Andrew Y. Ng and Michael. I. Jordan. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Proceedings of the Advances in Neural Information Processing Systems*.

Nathan Ratliff, J. Andrew Bagnell, and Martin A. Zinkevich. 2007. (online) subgradient methods for structured prediction. In *International Conference on Artificial Intelligence and Statistics*, March.

Yee Whye Teh. 2006. A bayesian interpretation of interpolated kneser-ney. Technical Report Technical Report TRA2/06, NUS School of Computing.

Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. 2005. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484.