Statistical Input Method based on a Phrase Class *n*-gram Model

Hirokuni Maeta[1] *Shinsuke Mori*[1]
(1) Graduate School of Informatics, Kyoto University, Yoshida Honmachi, Sakyo-ku, Kyoto, Japan
maeta@ar.media.kyoto-u.ac.jp, forest@i.kyoto-u.ac.jp

ABSTRACT
We propose a method to construct a phrase class *n*-gram model for Kana-Kanji Conversion by combining phrase and class methods. We use a word-pronunciation pair as the basic prediction unit of the language model. We compared the conversion accuracy and model size of a phrase class bi-gram model constructed by our method to a tri-gram model. The conversion accuracy was measured by F measure and model size was measured by the vocabulary size and the number of non-zero frequency entries. The F measure of our phrase class bi-gram model was 90.41%, while that of a word-pronunciation pair tri-gram model was 90.21%. In addition, the vocabulary size and the number of non-zero frequency entries in the phrase class bi-gram model were 5,550 and 206,978 respectively, while those of the tri-gram model were 22,801 and 645,996 respectively. Thus our method makes a smaller, more accurate language model.

KEYWORDS: Kana-Kanji Conversion, *n*-gram model, phrase-based model, class-based model.

Proceedings of the Second Workshop on Advances in Text Input Methods (WTIM 2), pages 1–14,
COLING 2012, Mumbai, December 2012.

1 Introduction

Japanese input methods are an essential technology for Japanese computing. Japanese has over 6,000 characters, which is much larger than the number of keys in a keyboard. It is impossible to map each Japanese character to a key. So an alternate input method for Japanese characters is needed. This is done by inputting a pronunciation sequence and converting it to an output sequence of words. Here, the input sequence of pronunciation is a sequence of *kana* characters and the output sequence of words is a mixture of *kana* and *kanji* characters. So this conversion is called *Kana-Kanji Conversion* (KKC).

The noisy channel model approach has been successfully applied to input methods (Chen and Lee, 2000; Mori et al., 1999). In KKC, a word sequence is predicted from a pronunciation sequence. The system is composed of two modules: a language model, which measures the likelihood of a word sequence in the language and a word-pronunciation model, which describes a relationship between a word sequence and a pronunciation sequence. Thus, the conversion accuracy of KKC depends on the language model and the word-pronunciation model. The language model is, however, more important since it describes the context and it is larger in size than the word-pronunciation model.

We focus on how to improve the language model for KKC. An n-gram model is generally used for many tasks. In KKC, considering the need for the conversion speed and the size of the language model, bi-gram models are often used. However, bi-gram models can not refer to a long history. A tri-gram model, which is also popular for many tasks, can refer a longer history but the size of tri-gram models is larger than that of bi-gram models. There have been many attempts at improving language models. A class n-gram model (Brown et al., 1992; Kneser and Ney, 1993; Mori et al., 1998), which groups words of similar behavior into a single class, and a phrase n-gram model (Deligne and Bimbot, 1995; Ries et al., 1996; Mori et al., 1997) which replaces some word sequences by single tokens, are known to be practical in speech recognition community.

In this paper, we propose a method to construct a smaller, more accurate language model for KKC. It is often thought that accurate models are larger and that small models are less accurate. However, we successfully built a smaller, more accurate language model. This is done by combining phrase and class methods. First, we collect phrases and construct a phrase sequence corpus. By changing the prediction unit from a word to a word sequence, the model can use a longer history. Then we perform word clustering to restrict the growth of the model size. As a result, we obtain a phrase class n-gram model. This model is small and expected to be accurate because it uses a history as long as those used by higher order n-gram models.

In order to test the effectiveness of our method, we compared the conversion accuracy and the model size of the phrase class bi-gram model constructed by our method to other language models. We used a word-pronunciation pair as the basic prediction unit of the language models. The conversion accuracy is measured by F measure, which is the harmonic mean of precision and recall. The F measure of our phrase class bi-gram model was 90.41%, while that of a word-pronunciation pair tri-gram model was 90.21%. In addition, the vocabulary size and the number of non-zero frequency entries in the phrase class bi-gram model were 5,550 and 206,978 respectively, while those of the tri-gram model were 22,801 and 645,996 respectively. These results show that our method of combining phrase and class methods makes a smaller, more accurate language model for KKC.

2 Statistical Input Method

In this section we give a brief explanation of a statistical input method and a word-pronunciation pair n-gram model, which is applied as the language model to KKC in the subsequent sections.

2.1 Input Method based on a Word n-gram Model

We explain a statistical approach to an input method based on the noisy channel model (Chen and Lee, 2000; Mori et al., 1999). This approach uses a word as the prediction unit of a language model.

Let $\boldsymbol{x} = x_1 x_2 \cdots x_l$ be a pronunciation sequence and $\boldsymbol{w} = w_1 w_2 \cdots w_m$ be a word sequence. Given $\boldsymbol{x}$ as input, the goal of the input method is to output $\hat{\boldsymbol{w}}$ that maximizes the probability $p(\boldsymbol{w}|\boldsymbol{x})$ as follows:

$$\hat{\boldsymbol{w}} = \operatorname*{\mathbf{argmax}}_{\boldsymbol{w}} p(\boldsymbol{w}|\boldsymbol{x}).$$

By Bayes' theorem,

$$p(\boldsymbol{w}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{x})}.$$

Since $p(\boldsymbol{x})$ is independent of $\boldsymbol{w}$, we have

$$\begin{aligned}
\hat{\boldsymbol{w}} &= \operatorname*{\mathbf{argmax}}_{\boldsymbol{w}} p(\boldsymbol{w}|\boldsymbol{x}) \\
&= \operatorname*{\mathbf{argmax}}_{\boldsymbol{w}} \frac{p(\boldsymbol{x}|\boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{x})} \\
&= \operatorname*{\mathbf{argmax}}_{\boldsymbol{w}} p(\boldsymbol{x}|\boldsymbol{w})p(\boldsymbol{w}).
\end{aligned}$$

Here, the problem is divided into two parts. $p(\boldsymbol{w})$ is a language model and we call $p(\boldsymbol{x}|\boldsymbol{w})$ a word-pronunciation model.

A language model $p(\boldsymbol{w})$ outputs the probability of a word sequence $\boldsymbol{w}$. First the probability $p(\boldsymbol{w})$ is assumed to be expressed as a product of a set of conditional probabilities

$$\begin{aligned}
p(\boldsymbol{w}) &= p(w_1 w_2 \cdots w_m) \\
&= \prod_{i=1}^{m} p(w_i|w_1 w_2 \cdots w_{i-1}). \qquad (1)
\end{aligned}$$

Then $p(w_i|w_1 \cdots w_{i-1})$ is approximated as k-order Markov process

$$p(w_i|w_1 w_2 \cdots w_{i-1}) \approx p(w_i|w_{i-k} w_{i-k+1} \cdots w_{i-1}),$$

where $k = n - 1$.

The other part, the word-pronunciation model $p(\boldsymbol{x}|\boldsymbol{w})$, outputs the probability of the pronunciation sequence $\boldsymbol{x}$ given the word sequence $\boldsymbol{w}$. This probability is assumed to be decomposed as follows:

$$p(\boldsymbol{x}|\boldsymbol{w}) = \prod_{i=1}^{m} p(\boldsymbol{x}_i|w_i),$$

where $\boldsymbol{x}_i$ is the sequence of pronunciation corresponding to the word w_i.

詰め/*tsu-me* 将棋/*sho-u-gi* の/*no* 本/*ho-n* を/*wo*
買/*ka* っ/*ttsu* て/*te* き/*ki* ま/*ma* し/*shi* た/*ta* 。/*.*

Figure 1: An example of a word-pronunciation pair sentence.

2.2 A Word-pronunciation Pair n-gram Model

In this paper we use a word-pronunciation pair n-gram model. A word-pronunciation pair n-gram model takes a pair of a word and its pronunciation as the prediction unit. Thus we can model a word and its pronunciation at the same time. This is because some Japanese kanji characters have the same pronunciation. So it is expected to be better to predict both a word and its pronunciation than predicting a word only.

Figure 1 shows an example of a corpus for training a word-pronunciation pair n-gram model. Units are separated with a white space. The left hand side of the slash in a unit is a word and right hand side is its pronunciation.

First we change the mathematics of the input method.

$$\begin{aligned} \hat{\boldsymbol{w}} &= \operatorname*{\mathbf{argmax}}_{\boldsymbol{w}} p(\boldsymbol{w}|\boldsymbol{x}) \\ &= \operatorname*{\mathbf{argmax}}_{\boldsymbol{w}} \frac{p(\boldsymbol{w},\boldsymbol{x})}{p(\boldsymbol{x})} \\ &= \operatorname*{\mathbf{argmax}}_{\boldsymbol{w}} p(\boldsymbol{w},\boldsymbol{x}). \end{aligned} \tag{2}$$

Here, the problem is $p(\boldsymbol{w},\boldsymbol{x})$ only. Then we express $p(\boldsymbol{w},\boldsymbol{x})$ by a word-pronunciation pair n-gram model as follows:

$$p(\boldsymbol{w},\boldsymbol{x}) = p(\langle w_1,\boldsymbol{x}_1\rangle\langle w_2,\boldsymbol{x}_2\rangle\cdots\langle w_m,\boldsymbol{x}_m\rangle),$$

where $\langle w_i,\boldsymbol{x}_i\rangle$ denotes a pair of a word w_i and its pronunciation $\boldsymbol{x}_i$. The character subsequences $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, ..., $\boldsymbol{x}_m$ satisfy that

$$\boldsymbol{x} = \boldsymbol{x}_1\boldsymbol{x}_2\cdots\boldsymbol{x}_m. \tag{3}$$

A word-pronunciation pair n-gram model outputs the probability of a word-pronunciation pair sequence $\langle w_1,\boldsymbol{x}_1\rangle\langle w_2,\boldsymbol{x}_2\rangle\cdots\langle w_m,\boldsymbol{x}_m\rangle$. Like word n-gram models, this probability is expressed as a product of a set of conditional probabilities and approximated as k-order Markov process, where $k = n-1$

$$\begin{aligned} &p(\langle w_1,\boldsymbol{x}_1\rangle\langle w_2,\boldsymbol{x}_2\rangle\cdots\langle w_m,\boldsymbol{x}_m\rangle) \\ &\quad = \prod_{i=1}^{m} p(\langle w_i,\boldsymbol{x}_i\rangle|\langle w_1,\boldsymbol{x}_1\rangle\langle w_2,\boldsymbol{x}_2\rangle\cdots\langle w_{i-1},\boldsymbol{x}_{i-1}\rangle), \\ &p(\langle w_i,\boldsymbol{x}_i\rangle|\langle w_1,\boldsymbol{x}_1\rangle\langle w_2,\boldsymbol{x}_2\rangle\cdots\langle w_{i-1},\boldsymbol{x}_{i-1}\rangle) \\ &\quad \approx p(\langle w_i,\boldsymbol{x}_i\rangle|\langle w_{i-k},\boldsymbol{x}_{i-k}\rangle\langle w_{i-k+1},\boldsymbol{x}_{i-k+1}\rangle\cdots\langle w_{i-1},\boldsymbol{x}_{i-1}\rangle). \end{aligned}$$

2.3 Parameter Estimation

The probability $p(\langle w_i, \boldsymbol{x}_i\rangle|\langle w_{i-k}, \boldsymbol{x}_{i-k}\rangle\langle w_{i-k+1}, \boldsymbol{x}_{i-k+1}\rangle\cdots\langle w_{i-1}, \boldsymbol{x}_{i-1}\rangle)$ is defined by maximum likelihood from a training corpus

$$\begin{aligned} &p(\langle w_i, \boldsymbol{x}_i\rangle|\langle w_{i-k}, \boldsymbol{x}_{i-k}\rangle\langle w_{i-k+1}, \boldsymbol{x}_{i-k+1}\rangle\cdots\langle w_{i-1}, \boldsymbol{x}_{i-1}\rangle) \\ &\quad \stackrel{def.}{=} \frac{N(\langle w_{i-k}, \boldsymbol{x}_{i-k}\rangle\langle w_{i-k+1}, \boldsymbol{x}_{i-k+1}\rangle\cdots\langle w_{i-1}, \boldsymbol{x}_{i-1}\rangle\langle w_i, \boldsymbol{x}_i\rangle)}{N(\langle w_{i-k}, \boldsymbol{x}_{i-k}\rangle\langle w_{i-k+1}, \boldsymbol{x}_{i-k+1}\rangle\cdots\langle w_{i-1}, \boldsymbol{x}_{i-1}\rangle)}, \end{aligned}$$

where $N(\langle w, \boldsymbol{x}\rangle\cdots)$ is the number of times that the word-pronunciation pair sequence occurs in the corpus. The word segmentation and pronunciation tagging of the training corpus should be accurate.

2.4 Interpolation

There are data-sparseness problems such that a zero-probability problem. In order to avoid these problems, we use a linear interpolation (Brown et al., 1992).

$$\begin{aligned} &p'(\langle w_i, \boldsymbol{x}_i\rangle|\langle w_{i-k}, \boldsymbol{x}_{i-k}\rangle\langle w_{i-k+1}, \boldsymbol{x}_{i-k+1}\rangle\cdots\langle w_{i-1}, \boldsymbol{x}_{i-1}\rangle) \\ &\quad = \sum_{j=0}^{k} \lambda_j p(\langle w_i, \boldsymbol{x}_i\rangle|\langle w_{i-j}, \boldsymbol{x}_{i-j}\rangle\langle w_{i-j+1}, \boldsymbol{x}_{i-j+1}\rangle\cdots\langle w_{i-1}, \boldsymbol{x}_{i-1}\rangle), \end{aligned}$$

where $0 \leq \lambda_j \leq 1$, $\sum_{j=0}^{k} \lambda_j = 1$.

2.5 Unknown Word Model

The probability $p(\boldsymbol{w}, \boldsymbol{x})$ in Equation (2) equals to 0 if the input character sequence $\boldsymbol{x}$ can not be expressed as a sequence of characters in the vocabulary. In other words, in this case, the system can not find $\langle w_1, \boldsymbol{x}_1\rangle\langle w_2, \boldsymbol{x}_2\rangle\cdots\langle w_m, \boldsymbol{x}_m\rangle$ that satisfies Equation (3). So we need a model that gives the probability larger than 0 for these character sequences. This model is called an unknown word model.

An unknown word model $p(\boldsymbol{x})$ outputs the probability of a character sequence $\boldsymbol{x}$ and it should give a higher probability if the character sequence is likely a word not in a vocabulary of the language model.

As word n-gram models are often used to predict a word sequence, character n-gram models are often used to predict a character sequence of an unknown word. So $p(\boldsymbol{x})$ is easily trained from the words in a training corpus that are not in the vocabulary.

3 Language Model Improvement

Class modeling (Brown et al., 1992; Kneser and Ney, 1993; Mori et al., 1998) makes smaller language models and phrase modeling (Deligne and Bimbot, 1995; Ries et al., 1996; Mori et al., 1997) makes more accurate language models. They are known to be practical in speech recognition community. In this section we give a brief explanation of these improvements. The prediction unit of the models here is a word, but we apply these ideas to the word-pronunciation pair n-gram model for our statistical input method by changing the prediction unit.

3.1 Class n-gram Model

We use an n-gram model as the language model for KKC. An n-gram model predicts the i-th word referring to the last $n-1$ words. Although an n-gram model works well, the model size is often the serious problem.

One of the solutions to this problem is grouping similar words together. By assigning words to its class, we can obtain the language model of less states. Let f be a function that maps a word w into its class c. Given the map f, a *class n-gram model* predicts the i-th class referring to the last $k = n-1$ classes and the i-th word referring to the i-th class. Therefore we define $p(w_i|w_{i-k}w_{i-k+1}\cdots w_{i-1})$ in Equation (1) as follows:

$$p(w_i|w_{i-k}w_{i-k+1}\cdots w_{i-1}) \stackrel{def.}{=} p(c_i|c_{i-k}c_{i-k+1}\cdots c_{i-1})p(w_i|c_i),$$

where $c_* = f(w_*)$. With the map f, $p(c_i|c_{i-k}c_{i-k+1}\cdots c_{i-1})$ and $p(w_i|c_i)$ are defined by maximum likelihood from a training corpus.

$$p(c_i|c_{i-k}c_{i-k+1}\cdots c_{i-1}) \stackrel{def.}{=} \frac{N(c_{i-k}c_{i-k+1}\cdots c_{i-1}c_i)}{N(c_{i-k}c_{i-k+1}\cdots c_{i-1})}$$

$$p(w_i|c_i) \stackrel{def.}{=} \frac{N(w_i)}{N(c_i)},$$

where $N(c\cdots)$ is the number of times that the class sequence occurs in the corpus.

The problem is how to determine the map f. In other words, how to do *word clustering*. Of course it should be done carefully, or the language model would get worse.

There are some methods for word clustering. Here we follow (Mori et al., 1998)'s clustering method, where the criterion is the average cross-entropy of sub corpora. Because of this criterion, this method is expected not to increase the cross-entropy of language models.

In the method, first the training corpus is split into k sub corpora C_1, C_2, ..., C_k. Then $\{C_1, C_2, \ldots, C_k\} \setminus \{C_i\}$ is used to estimate the model parameters and C_i is used to evaluate the objective function for $i = 1, 2, \cdots, k$. The objective function is the average cross-entropy

$$\overline{H}(f) \stackrel{def.}{=} \frac{1}{k}\sum_{i=1}^{k} H(M_i(f), C_i),$$

where $M_i(f)$ is a class n-gram model constructed from f, which maps a word into its class, and sub corpora $\{C_1, C_2, \ldots, C_k\} \setminus \{C_i\}$. Thus $H(M_i(f), C_i)$ is the cross-entropy of the corpus C_i by the model $M_i(f)$ and $\overline{H}(f)$ is their average. The lower $\overline{H}(f)$ is, the better f is expected to be. This is just like evaluating a language model.

Therefore the best map $\hat{f}$ is the one that minimizes $\overline{H}(f)$. Namely,

$$\hat{f} = \mathop{\mathbf{argmin}}_{f} \overline{H}(f).$$

It is, however, impossible to find the best map among all the possible maps. Alternatively we apply a greedy algorithm, which is shown in Figure 2, to find an optimal $\hat{f}$.

```
1: Sort words w_1, w_2, ..., w_N by frequency in descending order.
2: for i ← 1,N do
3:     c_i ← {w_i}
4:     f(w_i) ← c_i
5: end for
6: for i ← 2,N do
7:     c ← argmin_{c∈{c_1, c_2, ..., c_{i-1}}} H̄(MOVE(f, w_i, c))
8:     if H̄(MOVE(f, w_i, c)) < H̄(f) then
9:         f ← MOVE(f, w_i, c)
10:    end if
11: end for

12: procedure MOVE(f, w, c)
13:    f(w) ← f(w) \ {w}
14:    c ← c ∪ f(w)
15:    return f
16: end procedure
```

Figure 2: Clustering algorithm

3.2 Phrase n-gram Model

Here we explain the other language model improvement. We saw two kinds of n-gram models so far: a word n-gram model and a class n-gram model. Both models predict the next word referring to the last $n-1$ words. In other words, each of the prediction units of the models is a word. Actually, language models whose prediction unit is a word does not always have the best performance. A *phrase n-gram model*, whose prediction unit is a word sequence, can treat a word sequence that co-occurs frequently as a single word. Thus we can expect that it improves n-gram models.

The mathematics of a phrase n-gram model is the same as that of a word n-gram model except for the prediction unit because of the definition of the phrase model.

Let $\boldsymbol{w} = w_1 w_2 \cdots w_m$ be the input sequence of words. In a phrase n-gram model, the word sequence $\boldsymbol{w}$ is converted to the phrase sequence $\boldsymbol{\gamma} = \gamma_1 \gamma_2 \cdots \gamma_{m'}$ and the phrase sequence $\boldsymbol{\gamma}$ is predicted,

$$p(\boldsymbol{w}) \stackrel{def.}{=} p(\boldsymbol{\gamma}).$$

A phrase γ_i is a token that represents a word sequence. So $\boldsymbol{\gamma}$ becomes $\boldsymbol{w}$ if each phrase is replaced by its words.
Then $p(\boldsymbol{\gamma})$ is calculated like word n-gram models,

$$\begin{aligned} p(\boldsymbol{\gamma}) &= p(\gamma_1 \gamma_2 \cdots \gamma_{m'}) \\ &= \prod_{i=1}^{m'} p(\gamma_i | \gamma_1 \cdots \gamma_{i-1}), \\ p(\gamma_i | \gamma_1 \cdots \gamma_{i-1}) &\approx p(\gamma_i | \gamma_{i-k} \gamma_{i-k+1} \cdots \gamma_{i-1}), \end{aligned}$$

$\Gamma \leftarrow \{\}$
Estimate the interpolation coefficients.
Collect all the word sequences $\gamma_1, \gamma_2, \cdots, \gamma_i, \cdots$ where γ_i is a sequence of 2 or more words and it occurs in all the sub corpora C_1, C_2, …, C_k.
Sort $\gamma_1, \cdots, \gamma_N, \gamma_{N+1}, \cdots$ so that $\overline{H}(\{\gamma_1\}) < \cdots < \overline{H}(\{\gamma_N\}) < \overline{H}(\{\}) < \overline{H}(\{\gamma_{N+1}\})$.
for $i \leftarrow 1, N$ **do**
    if $\overline{H}(\Gamma \cup \{\gamma_i\}) < \overline{H}(\Gamma)$ **then**
        $\Gamma \leftarrow \Gamma \cup \{\gamma_i\}$
        Estimate the interpolation coefficients.
    end if
end for

Figure 3: Algorithm for finding phrases

where $k = n - 1$. $p(\gamma_i|\gamma_{i-k}\gamma_{i-k+1}\cdots\gamma_{i-1})$ is defined by maximum likelihood from a training corpus

$$p(\gamma_i|\gamma_{i-k}\gamma_{i-k+1}\cdots\gamma_{i-1}) \stackrel{def.}{=} \frac{N(\gamma_{i-k}\gamma_{i-k+1}\cdots\gamma_{i-1}\gamma_i)}{N(\gamma_{i-k}\gamma_{i-k+1}\cdots\gamma_{i-1})},$$

where $N(\gamma\cdots)$ is the number of times that the phrase sequence occurs in the corpus.

Phrase n-gram models require a phrase sequence corpus that can be obtained by collecting sequences of words for phrases and replacing each of those sequences in the original corpus by a single token. Sequences of words should be collected so that, by treating each of them as a single token, a phrase n-gram model has less cross-entropy.

Here we follow (Mori et al., 1997), where sequences of words are collected so that the average cross-entropy of sub corpora becomes less by the replacement of those sequences by tokens. This method find a sequence of greater than or equal to two words and it is expected to decrease the cross-entropy of language models because of the criterion.

The method of finding phrases is similar to that of word clustering. Here the training corpus is split into k sub corpora C_1, C_2, …, C_k. $\{C_1, C_2, \ldots, C_k\} \setminus \{C_i\}$ is used to estimate the model parameters and C_i is used to evaluate the objective function for $i = 1, 2, \cdots, k$. The objective function is the average cross-entropy

$$\overline{H}(\Gamma) \stackrel{def.}{=} \frac{1}{k}\sum_{i=1}^{k} H(M_i(\Gamma), C_i),$$

where $M_i(\Gamma)$ is a phrase n-gram model constructed from a set of word sequences for phrases Γ and sub corpora $\{C_1, C_2, \ldots, C_k\} \setminus \{C_i\}$. Thus $H(M_i(\Gamma), C_i)$ is the cross-entropy of the corpus C_i by the model $M_i(\Gamma)$ and $\overline{H}(\Gamma)$ is their average. The lower $\overline{H}(\Gamma)$ is, the better Γ is expected to be.

Therefore the best set of word sequences $\hat{\Gamma}$ is the one that minimizes $\overline{H}(\Gamma)$. Namely,

$$\hat{\Gamma} = \mathop{\mathbf{argmin}}_{\Gamma} \overline{H}(\Gamma).$$

It is, however, impossible to find the best set among all the possible sets. Alternatively we apply a greedy algorithm, which is shown in Figure 3, to find an optimal $\hat{\Gamma}$.

1: For the given training corpus, find the optimal set of word sequences $\hat{\Gamma}$ using the algorithm in Figure 3.
2: Build the phrase sequence corpus from $\hat{\Gamma}$ acquired in step 1 by replacing the word sequence with its phrase.
3: For this phrase sequence corpus, find the optimal map $\hat{f}$ using the algorithm in Figure 2.

Figure 4: Combining the phrase and class methods.

4 Statistical Input Method based on a Phrase Class n-gram Model

In this section, we propose a method of combining the two language model improvements in the section 3 and construct a *phrase class n-gram model* so that we can realize the compact but accurate system. In addition, we also explain how to integrate the phrase class n-gram model into the statistical input method.

4.1 Combining the Phrase and Class Methods

A phrase model has less cross-entropy than a word model (Mori et al., 1997). Phrase modeling makes a more accurate language model. However the vocabulary size of phrase models is larger than that of word models, because some word sequences for phrases are added to the vocabulary of the phrase models. As a result, phrase models are often larger than word models.

This problem is solved by the method of class modeling. A class model is smaller than a word model (Brown et al., 1992; Kneser and Ney, 1993; Mori et al., 1998). A vocabulary size of a phrase model can be decreased by clustering the words and phrases in the vocabulary using the algorithm in Figure 2. Figure 4 shows our procedure of combining the two methods. As shown in this procedure, the clustering algorithm is applied to the phrase sequence corpus used for a phrase model.

4.2 Phrase Class n-gram Model

We can construct a phrase class n-gram model using a phrase sequence corpus and a function f that maps a phrase into its class. Because the phrase class n-gram model is made by combining the phrase and class methods, the mathematics of the phrase class n-gram model is also the combination of the mathematics of the two models.

Let $\boldsymbol{w} = w_1 w_2 \cdots w_m$ be an input sequence of words and f be a function that maps a phrase γ into its class c. In a phrase class model, like phrase models, $\boldsymbol{w}$ is converted to the phrase sequence $\boldsymbol{\gamma} = \gamma_1 \gamma_2 \cdots \gamma_{m'}$, which is corresponding to $\boldsymbol{w}$, and the phrase sequence $\boldsymbol{\gamma}$ is predicted.

$$
\begin{aligned}
p(\boldsymbol{w}) &\stackrel{def.}{=} p(\boldsymbol{\gamma}) \\
&= p(\gamma_1 \gamma_2 \cdots \gamma_{m'}) \\
&= \prod_{i=1}^{m'} p(\gamma_i | \gamma_1 \gamma_2 \cdots \gamma_{i-1}), \\
p(\gamma_i | \gamma_1 \gamma_2 \cdots \gamma_{i-1}) &\approx p(\gamma_i | \gamma_{i-k} \gamma_{i-k+1} \cdots \gamma_{i-1}).
\end{aligned}
$$

Then, like class models, the phrase class n-gram model predicts the i-th class referring to the

last $k = n-1$ classes and the i-th phrase referring to the i-th class.

$$p(\gamma_i|\gamma_{i-k}\gamma_{i-k+1}\cdots\gamma_{i-1}) \stackrel{def.}{=} p(c_i|c_{i-k}c_{i-k+1}\cdots c_{i-1})p(\gamma_i|c_i), \tag{4}$$

where $c_* = f(\gamma_*)$. $p(c_i|c_{i-k}c_{i-k+1}\cdots c_{i-1})$ and $p(\gamma_i|c_i)$ are defined by maximum likelihood from a training corpus.

$$p(c_i|c_{i-k}c_{i-k+1}\cdots c_{i-1}) \stackrel{def.}{=} \frac{N(c_{i-k}c_{i-k+1}\cdots c_{i-1}c_i)}{N(c_{i-k}c_{i-k+1}\cdots c_{i-1})}, \tag{5}$$

$$p(\gamma_i|c_i) \stackrel{def.}{=} \frac{N(\gamma_i)}{N(c_i)}. \tag{6}$$

The prediction unit of the phrase class n-gram model constructed here is a word. We can construct in the same way a phrase class n-gram model whose basic prediction unit is a word-pronunciation pair, and it can be applied to the input method in the section 2.2. In that case, γ_* in this section represents a word-pronunciation pair sequence.

4.3 Input Method based on a Phrase Class n-gram Model

We integrate the phrase class n-gram model, where the basic prediction unit is a word-pronunciation pair, and the unknown word model described in the section 2.5.

Given a pronunciation sequence $\boldsymbol{x}$ as an input, the goal of the input method is to output $\hat{\boldsymbol{w}}$ that maximizes the likelihood $p(\boldsymbol{w},\boldsymbol{x}) = p(\langle w_1,\boldsymbol{x}_1\rangle\langle w_2,\boldsymbol{x}_2\rangle\cdots\langle w_m,\boldsymbol{x}_m\rangle)$, as we explained in the section 2.2. We use a phrase class n-gram model whose basic prediction unit is a word-pronunciation pair to calculate $p(\langle w_1,\boldsymbol{x}_1\rangle\langle w_2,\boldsymbol{x}_2\rangle\cdots\langle w_m,\boldsymbol{x}_m\rangle)$. In the phrase class n-gram model, first $\langle w_1,\boldsymbol{x}_1\rangle\langle w_2,\boldsymbol{x}_2\rangle\cdots\langle w_m,\boldsymbol{x}_m\rangle$ is converted to the phrase sequence $\boldsymbol{\gamma} = \gamma_1\gamma_2\cdots\gamma_{m'}$, where the phrase γ_i represents a word-pronunciation pair sequence, and then $p(\boldsymbol{\gamma})$ is calculated as follows:

$$\begin{aligned}
p(\langle w_1,\boldsymbol{x}_1\rangle\langle w_2,\boldsymbol{x}_2\rangle\cdots\langle w_m,\boldsymbol{x}_m\rangle) &\stackrel{def.}{=} p(\boldsymbol{\gamma}) \\
&= p(\gamma_1\gamma_2\cdots\gamma_{m'}) \\
&= \prod_{i=1}^{m'} p(\gamma_i|\gamma_1\gamma_2\cdots\gamma_{i-1}), \\
p(\gamma_i|\gamma_1\gamma_2\cdots\gamma_{i-1}) &\approx p(\gamma_i|\gamma_{i-k}\gamma_{i-k+1}\cdots\gamma_{i-1}).
\end{aligned}$$

If γ_i is in the vocabulary of the phrase class n-gram model, by Equations (4), (5) and (6) we have

$$p(\gamma_i|\gamma_{i-k}\gamma_{i-k+1}\cdots\gamma_{i-1}) = \frac{N(c_{i-k}c_{i-k+1}\cdots c_{i-1}c_i)}{N(c_{i-k}c_{i-k+1}\cdots c_{i-1})}\frac{N(\gamma_i)}{N(c_i)}.$$

Otherwise, we approximate $p(\gamma_i|\gamma_{i-k}\gamma_{i-k+1}\cdots\gamma_{i-1})$ using the unknown word model $p(\boldsymbol{x})$ in the section 2.5 as follows:

$$p(\gamma_i|\gamma_{i-k}\gamma_{i-k+1}\cdots\gamma_{i-1}) \approx p(\boldsymbol{x'}_i),$$

where $\boldsymbol{x'}_i$ is the character sequence corresponding to γ_i.

usage	#sentences	#words	#chars
training	53,424	1,258,805	1,813,135
test	6,350	137,642	201,477

Table 1: Corpora.

5 Experiments

In order to test the effectiveness of combining the phrase and class methods, we constructed phrase class bi-gram using our method and compared the conversion accuracy and the model size of our model to other language models. In this section we show the experimental results and discuss them.

5.1 Experimental Settings

We used the core corpus of the Balanced Corpus of Contemporary Written Japanese (BCCWJ) (Maekawa, 2008) for the experiments. The BCCWJ is composed of two corpora, the BCCWJ-CORE corpus and the BCCWJ-NON-CORE corpus, and we used the BCCWJ-CORE. A sentence of the BCCWJ-CORE corpus is a sequence of pairs of a word and its pronunciation as shown in Figure 1. The word segmentation and pronunciation tagging were done manually.

We split the BCCWJ-CORE into two parts: one is for training language models and the other is for the test of them. Table 1 shows the specifications of the corpora.

We constructed language models and a vocabulary from the training corpus of the BCCWJ. The vocabulary was constructed according to (Mori et al., 1998) and (Mori et al., 1997).

The following is a list of language models we compared.

1. Word-pronunciation pair bi-gram model
2. Class bi-gram model (Prediction unit: Word-pronunciation pair)
3. Phrase bi-gram model
4. Phrase class bi-gram model (Prediction unit: Phrase)
5. Word-pronunciation pair tri-gram model

5.2 Criterions

One of the criterions we took in this paper is conversion accuracy. The conversion accuracy is measured by F measure, which is the harmonic mean of precision P and recall R. Let N_{CWJ} be the number of characters of a sentence in the BCCWJ for the test, N_{SYS} be the number of characters of a sentence that KKC outputs, and N_{LCS} be the number of characters of the longest common subsequence. The precision P is defined as N_{LCS}/N_{SYS} and the recall R is defined as N_{LCS}/N_{CWJ}. Hence F measure is $\frac{2}{1/P+1/R}$.

In addition to the conversion accuracy, we also evaluated our methods from the viewpoint of the size of language models. The size of a language model is measured by the vocabulary size and the number of non-zero frequency entries of bi-gram or tri-gram.

Model	Precision	Recall	F measure
Word-pronunciation pair bi-gram	89.86	90.32	90.09
Class bi-gram	89.78	90.28	90.03
Phrase bi-gram	90.26	90.53	90.39
Phrase class bi-gram	90.25	90.58	90.41
Word-pronunciation pair tri-gram	89.97	90.45	90.21

Table 2: Conversion accuracy

Model	Vocabulary size	#non-zero frequency entries
Word-pronunciation pair bi-gram	22,801	264,336
Class bi-gram	4,245	141,482
Phrase bi-gram	25,056	339,574
Phrase class bi-gram	5,550	206,978
Word-pronunciation pair tri-gram	22,801	645,996

Table 3: Model size

5.3 Results

Table 2 shows the conversion accuracies of the bi-gram models and the tri-gram model. Comparing the conversion accuracies of the word-pronunciation pair bi-gram model and the phrase bi-gram model, it can be said that phrase modeling makes better language models. Class modeling does not have a significant effect on the conversion accuracy by comparing the conversion accuracies of the word-pronunciation pair bi-gram model and the class bi-gram model. The phrase class model, which was constructed by our method, has the best F measure.

Table 3 shows the model sizes of the bi-gram models and the tri-gram model. The vocabulary size of the phrase bi-gram model is larger than that of the word-pronunciation pair bi-gram model. Generally, the vocabulary size of a phrase model is larger than that of a word model since the phrase model adds some phrases to its vocabulary. We see that word clustering or phrase clustering makes smaller language models. As the result of phrase clustering, the size of our phrase class bi-gram model is smaller than that of the word-pronunciation pair bi-gram model and the word-pronunciation pair tri-gram model.

5.4 Evaluation on Large Training Data

We performed another more practical experiment. In this experiment, we also constructed a phrase class bi-gram model and a word-pronunciation pair tri-gram model. We used the training corpus that has about 360,000 sentences mainly from BCCWJ-NON-CORE. The results are in table 4 and 5. We see that the phrase class bi-gram model is smaller and it has the comparable conversion accuracy as the word-pronunciation pair tri-gram model.

6 Conclusion

In this paper we proposed a method to improve the language model for KKC by combining the phrase and class methods. We used a word-pronunciation pair as the basic prediction unit of the language model and constructed a phrase class n-gram model. We compared the conversion accuracy and model size of the phrase class bi-gram model to the other models. The results showed that the phrase class bi-gram model is smaller and it has the comparable or better

Model	Precision	Recall	F measure
Phrase class bi-gram	91.38	91.80	91.59
Word-pronunciation pair tri-gram	91.23	91.80	91.51

Table 4: Conversion accuracy

Model	Vocabulary size	#non-zero frequency entries
Phrase class bi-gram	10,421	862,890
Word-pronunciation pair tri-gram	61,040	3,225,937

Table 5: Model size

conversion accuracy than that of the word-pronunciation pair tri-gram model. Therefore our method of combining the improvements of phrase and class modeling makes a smaller, more accurate language model for KKC.

Acknowledgments

This work was partially supported by Microsoft CORE 8 Project.

References

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

Chen, Z. and Lee, K.-F. (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.

Deligne, S. and Bimbot, F. (1995). Language modeling by variable length sequences: Theoretical formulation and evaluation of multigrams. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 169–172.

Kneser, R. and Ney, H. (1993). Improved clustering techniques for class-based statistical language modelling. In *Proceedings of the Third European Conference on Speech Communication and Technology*, pages 973–976.

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

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

Mori, S., Nishimura, M., and Itoh, N. (1998). Word clustering for a word bi-gram model. In *Proceedings of the Fifth International Conference on Speech and Language Processing*.

Mori, S., Yamaji, O., and Nagao, M. (1997). An improvement of *n*-gram model by a change of the prediction unit. In *IPSJ SIG Technical Reports*, 97-SLP-19-14, pages 87–94. (In Japanese).

Ries, K., Buø, F. D., and Waibel, A. (1996). Class phrase models for language modeling. In *Proceedings of the Fourth International Conference on Speech and Language Processing*.