Using Collocations and K-means Clustering to Improve the N-pos Model for Japanese IME

Long Chen *Xianchao Wu* *Jingzhou He*
Baidu Inc. (Shenzhen and Tokyo)
{chenlong05,wuxianchao,hejingzhou}@baidu.com

Abstract
Kana-Kanji conversion is known as one of the representative applications of Natural Language Processing (NLP) for the Japanese language. The N-pos model, presenting the probability of a Kanji candidate sequence by the product of bi-gram Part-of-Speech (POS) probabilities and POS-to-word emission probabilities, has been successfully applied in a number of well-known Japanese Input Method Editor (IME) systems. However, since N-pos model is an approximation of n-gram word-based language model, important word-to-word collocation information are lost during this compression and lead to a drop of the conversion accuracies. In order to overcome this problem, we propose ways to improve current N-pos model. One way is to append the high-frequency collocations and the other way is to sub-categorize the huge POS sets to make them more representative. Experiments on large-scale data verified our proposals.

Keywords: Input Method Editor, K-means clustering, n-gram language model, collocation.

1 Introduction

In Japanese IME systems [1], Kana-Kanji conversion is known as one of the representative applications of NLP. Unfortunately, numerous researchers have taken it for granted that current NLP technologies have already given a fully support to this task and there are few things left to be done as research topics. However, as we go deeper to this "trivial" task, we recognize that converting from a romanized Hirakana sequence (i.e., users' input) into a mixture of Kana and Kanji sequence (i.e., users' expected output) is more difficult than it looks. Concretely, we are facing a lot of NLP research topics such as Japanese word/chunk segmentation, POS tagging, n-best decoding, etc. Existing algorithms dealing with these topics are challenged by the daily-updating and large-scale Web data.

Traditional n-gram word-level language model (short for "word n-gram model", hereafter) is good at ranking the Kanji candidates. However, by using the large-scale Web data in tera-bytes (TB), even bi-gram word-level language model is too large[2] to fit the memories (for loading the model) and computing abilities (for decoding) of users' personal computers (PCs). Dealing with this limitation, n-pos model (Kudo et al., 2011) was proposed to make a compression[3] of the word n-gram model. N-pos model takes POS tags (or word classes) as the latent variable and factorizes the probability of a Kanji candidate sequence into a product of POS-to-POS transition probabilities and POS-to-word

[1] The Japanese IME mentioned in this paper can be freely downloaded from: http://ime.baidu.jp/type/?source=pstop

[2] For example, using the 2.5TB data, the word n-gram model has 421 million 1-grams and 2.6 billion 2-grams.

[3] Indeed, as pointed out by one reviewer, n-pos model has its own benefits by organizing the semantically similar words and dealing with low frequency words. Thus, even the result n-pos model is smaller than word n-gram model, it is considered to be also bring accuracy improvements (Kneser and Ney, 1993; Mori et al.).

Proceedings of the Second Workshop on Advances in Text Input Methods (WTIM 2), pages 45–56,
COLING 2012, Mumbai, December 2012.

emission probabilities. This factorization makes n-pos model alike the well-known Hidden Markov Model (HMM). Since the number of POS tags is far smaller than the number of word types in the training data, n-pos model can significantly smaller the final model without deteriorating the conversion accuracies too much.

Compared with word n-gram model, n-pos model is good for its small size for both storing and decoding. However, the major disadvantage is that important word-level collocation information are not guaranteed to be kept in the model. One direction to remedy the n-pos model is to find those lost word-level information and append it. The other direction is to sub-categorize the original POS tags to make the entries under one POS tag contain as less homophonic words as possible. These considerations yielded our proposals, first by appending collocations and second by sub-categorizing POS tags. Experiments by making use of large-scale training data verified the effectiveness of our proposals.

This paper is organized as follows. In the next section, we give the formal definition of n-pos model and explain its disadvantage by real examples. In Section 3 we describe our proposed approaches. Experiments in Section 4 testify our proposals. We finally conclude this paper in Section 5.

2 N-pos Model and Its Disadvantage

2.1 N-pos model

For statistical Kana-Kanji conversion, we use $\mathbf{x}$ to express the input Hirakana sequence, $\mathbf{y}$ to express the output mixed Kana-Kanji sequence and $P(\mathbf{y}|\mathbf{x})$ to express the conditional probability for predicting $\mathbf{y}$ given $\mathbf{x}$. We further use $\hat{\mathbf{y}}$ to express the optimal $\mathbf{y}$ that maximize $P(\mathbf{y}|\mathbf{x})$ given $\mathbf{x}$. Based on the Bayesian theorem, we can derive $P(\mathbf{y}|\mathbf{x})$ from the product of the language model $P(\mathbf{y})$ and the Kanji-Kana (pronunciation) model $P(\mathbf{x}|\mathbf{y})$. This definition is also similar with that described in (Mori et al., 1999; Komachi et al., 2008; Kudo et al., 2011).

$$\begin{aligned}\hat{\mathbf{y}} &= \text{argmax} P(\mathbf{y}|\mathbf{x}) \\ &= \text{argmax} P(\mathbf{y}) P(\mathbf{x}|\mathbf{y})\end{aligned}$$

There are flexibilities in implementing the language model and the pronunciation model. Suppose the output $\mathbf{y}$ contains n words, i.e., $\mathbf{y} = w_1 ... w_n$. We use 1) product of word class bigram model and word class to word emission model as the language model, and 2) word-pronunciation unigram model as the Kanji-Kana model. That is,

$$P(\mathbf{y}) = \prod_{i=1}^{n} P(w_i|c_i) P(c_i|c_{i-1}) \tag{1}$$

$$P(\mathbf{x}|\mathbf{y}) = \prod_{i=1}^{n} P(r_i|w_i) \tag{2}$$

Here, c_i is the word class for word w_i (frequently corresponds to POS tags or inflected forms), $P(w_i|c_i)$ is the word generation probability from a word class c_i to word w_i, $P(c_i|c_{i-1})$ is the word class transition probability, r_i is the Kana pronunciation candidate for word w_i, and $P(r_i|w_i)$ is the probability that word w_i is pronounced as r_i. The optimal output $\hat{\mathbf{y}}$ (or even n-best list) can be effectively computed by the Viterbi algorithm (Viterbi, 1967; Huang and Chiang, 2005).

There are many methods for designing the word classes, such as unsupervised clustering, POS tags, etc. Following (Kudo et al., 2011), we designed the word classes by using the POS information

generated by an open-source toolkit Mecab[4] (Kudo et al., 2004) which was developed for Japanese word segmenting and POS tagging. Since POS bi-gram model plays an essential role in Equation 1, we call it *n-pos model*. Specially, similar to (Kudo et al., 2011), we also use the following rules to determine a word class:

- the deepest POS tag layers (totally six layers) of the IPA POS system[5] was used;
- for the words with inflection forms, their conjugated forms and inflections are all appended;
- particles, auxiliary verbs, and non-independence content words[6] are all taken as independent word classes; and,
- high-frequency verbs, nouns except named entities, adjectives, suffixes, prefixes are all taken as independent word classes.

Since there are many special words that are taken as word classes, we finally obtained around 2,500 word classes.

Probabilities $P(w_i|c_i)$ and $P(c_i|c_{i-1})$ can be computed from the POS-tagged corpus by using the maximum likelihood estimation method. The Kanji-Kana pronunciation model $P(r_i|w_i)$ can be computed by first mining Kanji-Kana pairs from the Web and then estimate their probabilities in a maximum likelihood way. Since Mecab also assigns Kana pronunciations and POS tags to Japanese words simultaneously during performing word segmentation, we can simply estimate $P(r_i|w_i)$ using the Web corpus pre-processed by Mecab. That is, $P(r_i|w_i) = freq(r_i, w_i)/freq(w_i)$. Here, function $freq()$ returns the (co-)frequency of a word and/or a Kana sequence in the training data. In our IME system, besides our basic Kana-Kanji conversion dictionary, the Kanji-Kana pairs mined from the Web are individually taken as "cell dictionaries". That is, they are organized by their category such as “idioms", “actor names", and so on. These cell dictionaries are optimal to the users and they can download those dictionaries which fit their interests. We also use a log-style function based on the frequencies of the Kanji candidates to compute the weights of the Kana-Kanji pairs. The weights are used to determine the rank of the Kanji candidates to be shown to the users.

2.2 The disadvantage

The basic motivation for factorizing $P(\mathbf{y})$ into Equation 1 is to compress the word n-gram model into the production of a bigram n-pos model $P(c_i|c_{i-1})$ and an emission model $P(w_i|c_i)$. N-pos model is good for its small size and the usage of syntactic information for predicting the next word. However, compared with word n-gram model, the disadvantage is clear: the co-frequency information of w_{i-1} and w_i is not taken into consideration during predicting.

Figure 1 shows an example for intuitive understanding of the disadvantage. Suppose both w_{i-1} (gennshi/nuclear) and w_i (hatsudenn/generate electricity) are low-frequency words in the training corpus, yet w_{i-1} always appears together with w_i (or we say w_{i-1} and w_i form a collocation). Under n-pos model, the total score of w_{i-1} w_i is determined mainly by $P(c_i|c_{i-1})$ and $P(w_i|c_i)$, but not $P(w_i|w_{i-1})$. Thus, the best candidate “nuclear power” in word n-gram model is possibly not be able to be predicated as the top-1 candidate in n-pos model.

[4]http://mecab.sourceforge.net/

[5]http://sourceforge.jp/projects/ipadic/

[6]non-independent content words (such as oru, aru, etc.) are those content words that do not have an independent semantic meaning, but have to be used together with another independent content word to form a complete meaning.

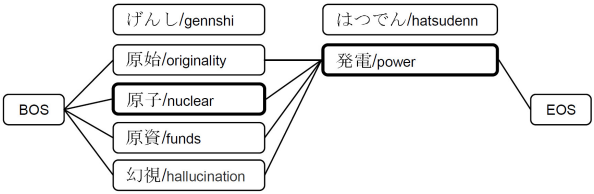

Figure 1: An example for the disadvantage of n-pos model.

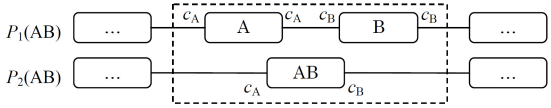

Figure 2: Changing n-pos model by replacing individual words A and B with collocation AB.

3 The Proposed Method

3.1 Appending "partial" word n-gram model

The disadvantage of n-pos model is mainly caused by its compression of word n-gram model. N-pos model can deal with a large part of the Kana-Kanji conversion problem yet short at dealing with collocations. We are wondering if partial of the word n-gram model can be "appended" to the n-pos model to further improve the final conversion precision.

The challenge is how to balance the usage of the two models for ranking the Kanji candidates. The score of a Kanji candidate sequence AB (with two words) can be computed by both the word n-gram model and the n-pos model. One simple consideration is to trust word n-gram model whenever n-pos model "fails" to make a better ranking. That is, we make use of word n-gram model only if candidate AB was assigned a higher score in word n-gram model than that in n-pos model.

We explain this idea through an example shown in Figure 2. In this figure, we want to replace individual words A and B in the decoding word lattice in the original n-pos model by a collocation AB, knowing that A and B sharing a high co-frequency in the training data.

$$
\begin{aligned}
P_1(AB) &= P(A|c_A)P(c_B|c_A)P(B|c_B) \\
&= \frac{freq(A)}{freq(c_A)} \times \frac{freq(c_Ac_B)}{freq(c_A)} \times \frac{freq(B)}{freq(c_B)}; \\
P_2(AB) &= \frac{freq(AB)}{freq(c_A)}.
\end{aligned}
$$

Here, c_A and c_B stand for the POS tag (or word class) of word A and B; function $freq()$ returns the frequency of words and POS tags in the training data. When appending collocations to the n-pos model, we need to let $P_1(AB) < P_2(AB)$ to ensure the collocation candidate has a higher rank in the candidate set. That is,

$$\frac{freq(A)}{freq(c_A)} \times \frac{freq(c_B)}{freq(c_A)} \times \frac{freq(B)}{freq(c_B)} < \frac{freq(AB)}{freq(c_A)}, i.e.,$$
$$\frac{freq(A)freq(B)}{freq(c_A)freq(c_B)} < \frac{freq(AB)}{freq(c_Ac_B)} \quad (3)$$

We make use of Formula 3 for mining collocations from the bi-grams in the training data.

There is one approximation in Formula 3. For collocation AB, its word class sequence is c_Ac_B. When computing $P_2(AB)$, we only used $freq(c_A)$ instead of $freq(c_Ac_B)$. Note that when computing the right-hand-side of AB in the second line, we still use c_B as the right-hand-side POS of AB. Similar strategy (of using c_A as the left-hand-side POS tag and c_B as the right-hand-side POS tag of AB) has been applied in Mecab and ChaSen[7] for Japanese word segmenting and POS tagging.

3.2 K-means clustering

The objective for unsupervised K-means clustering is to avoid (as much as possible) assigning entries with identical pronunciations or with large frequency variances into one word class. One big problem that hurts the precision of n-pos model is the existence of Kanji candidates with identical pronunciations in one word class, since the ranking is only determined by $p(w_i|c_i)$. If we could assign different word classes to the homophonic Kanji candidates, we can further make use of $P(c_i|c_{i-1})$ instead of only $P(w_i|c_i)$ to yield a better candidate ranking.

We define a kernel function[8] $F(A_1,A_2)$ to describe the difference of pronunciations between two Kanji candidates A_1 and A_2:

$$F(A_1,A_2) = \frac{1}{ed(pron(A_1),pron(A_2)) + 0.001} \quad (4)$$

Here, function $pron()$ returns the romanized Japanese Katakana sequence of a Kanji word[9]; function $ed()$ returns the edit distance of two string parameters. In this paper, we use the Levenshtein distance as defined in (Levenshtein, 1966). Through this definition, we know that the smaller the edit distance is, the larger value the $F()$ function returns. In case of A_1 and A_2 share an identical pronunciation, $F()$ returns the maximum value of 1,000. On the other hand, the bigger the edit distance, $F()$ is closer to 0. Thus, we say $F() \in (0, 1000]$.

Table 1 shows the top-5 high frequency pronunciations in our Kana-Kanji conversion lexicon. In this table, *yuuki* takes the highest frequency of 156. We thus set the K in K-means clustering to be 156 so that optimally all the 156 Kanji candidates with the pronunciation of *yuuki* can be assigned to some word class that is different from each other.

During K-means clustering, we first randomly pack 156 pronunciations as centre points from the Kana-Kanji conversion lexicon and then computer the distance score of $F()$ by using Equation 4. We aggressively assign one entry to the word class with the biggest $F()$ score.

[7] http://chasen-legacy.sourceforge.jp/

[8] As pointed out by one reviewer, this is not the only way to estimate the distance between to entries. Indeed, there are many personal names which should be taken as one special word class, we take this as an interesting future work.

[9] For example, in Figure 1, "gennshihatsudenn" is the result by applying $pron()$ to the Kanji candidates.

Pronunciation	Frequency	Kanji examples
ユウキ/yuuki	156	雄喜　裕希　裕記　雄生　勇企
コウジ/kouji	149	紘治　甲児　小池　講じ　小路
コウキ/kouki	134	弘毅　興起　晃毅　幸喜　功喜
ヨシヒロ/yoshihiro	118	義洋　佳洋　愛弘　吉広　善廣
ヒロシ/hiroshi	102	広し　洋資　博至　啓　博四

Table 1: The top-5 high frequency pronunciations in our Kana-Kanji conversion lexicon.

Rank	Six level POS tags	#	%	Top-Kana freq.	Top-Kana
1	名詞 一般 * * * *	359,951	39.69%	28	コウキ/kouki
2	名詞 固有名詞 一般 * * *	78,752	8.68%	7	アイノウタ/ainouta
3	名詞 固有名詞 地域 一般 * *	75,190	8.29%	11	コウヨウ/kouyou
4	名詞 固有名詞 人名 名 * *	61,280	6.76%	132	ユウキ/yuuki
5	名詞 固有名詞 組織 * * *	60,489	6.67%	7	セイコウ/seikou
6	名詞 固有名詞 人名 一般 * *	49,704	5.48%	6	カトウヒロシ/katouhiroshi
7	名詞 固有名詞 人名 姓 * *	31,649	3.49%	14	ミナミ/minami
8	名詞 サ変接続 * * * *	12,919	1.41%	15	センコウ/sennkou
9	動詞 自立 * * 一段 連用形	6,177	0.68%	11	カケ/kake
10	動詞 自立 * * 一段 未然形	6,174	0.68%	11	カケ/kake
			81.83%	242	

Table 2: The top-10 high frequency six-level POS tags in our Kana-Kanji conversion lexicon.

There is one difficulty that we should mention here. Note that the edit distance function does not satisfies *triangle inequality*. That is, we cannot ensure $ed(A,B) < ed(A,C) + ed(C,B)$. This makes it a bit difficult to determine the new centre point in a word class. In our approach, instead of drawing the centre point by a determined string for the future computing of $F()$ for a given pronunciation string, we use the averaged $F()$ score from the given string to all the string in a word class as the *distance*. This modification changes the complexity of K-means clustering algorithm from $O(nKt)$ to $O(n^2Kt)$, where n is the number of entries in the Kana-Kanji conversion lexicon, K is the number of word classes, and t is the number of iterations.

In our preliminary experiments, we found that clustering on the whole conversion lexicon did not yield a better result. The major reason was that, we ignored the frequency information of the entries, POS tags, and pronunciations in the lexicon. That is, we should make a sub-categorization on the big POS sets, such as nouns, verbs, adjectives.

Table 2 lists the top-10 high frequency six-level POS tags[10] in our Kana-Kanji conversion lexicon which contains 907,003 entries. Note that the top-8 POS tags are nouns and occurs 80.47% of the lexicon. Table 2 also lists the most frequently appears Kana (i.e., pronunciation) in each of the POS set. If we independently run K-means clustering for each of the top-10 POS sets and take K to be the number of the top-Kata frequency, we will extend these 10 sets into 242 sets.

$$J = \sum_{i=1}^{K} \sum_{p \in c_i} \frac{\sum_{p' \in c_i} F^2(p,p')}{|c_i|} \tag{5}$$

[10]http://sourceforge.jp/projects/ipadic/

The objective function of the K-means clustering algorithm is shown in Formula 5. Here, c_i represents a word class, p and p' are word entries (with word, POS tag, and pronunciation) in c_i, and $F()$ function is defined in Equation 4.

4 Experiments

4.1 Setup

As mentioned earlier, we use 2.5TB Japanese Web pages as our training data. We run Mecab on Hadoop[11], an open source software that implemented the Map-Reduce framework (Dean and Ghemawat, 2004), for word segmenting and POS tagging the data. Then, based on maximum likelihood estimation, we estimate $P(c_i|c_{i-1})$, $P(w_i|c_i)$, and $P(r_i|w_i)$ (referring to Equation 1 and 2). Our foundational Kana-Kanji conversion lexicon contains 907,003 entries. Based on the re-constructing strategies described in Section 2.1, we initially obtained 2,569 word classes for these lexicon entries.

We report conversion accuracies on three test sets:

- 23Kw: this test set contains 23K common words that are manually collected from the Web (w is short for “word" level test set);
- 6Ks: this test set contains 6K sentences that are randomly collected from the Web as well (s is short for “sentence" level test set);
- 5Kw: this test set contains 5K words that are manually collected from the Web.

Specially, the 5K test set includes the following entries:

- 2.5K high frequency words that are collected from the Web;
- 1K common words that are randomly selected from Nagoya University’s common word list[12];
- 0.5K basic concept verbs;
- 0.2K single Bensetsu (alike English chunk) words that are manually collected from the Web;
- 0.2K Japanese family names;
- 0.2K Japanese idioms;
- 0.2K Japanese place names;
- 0.2K Japanese single Kanji characters.

We use the following evaluation metrics:

- top-1/3/5 “precision”, i.e., if the reference Kanji string is included in the 1(3/5)-best output list;

[11] http://hadoop.apache.org/
[12] http://kotoba.nuee.nagoya-u.ac.jp/jc2/base/list

Test set	System	Top-1	Top-3	Top-5	1st screen	Recall
23Kw	baseline	73.34%	90.30%	94.08%	96.75%	98.71%
23Kw	+collocations	**73.48%**	**90.58%**	94.23%	**96.91%**	**98.87%**
23Kw	+clustering	73.30%	90.57%	**94.24%**	96.86%	98.76%
23Kw	+collocations+clustering	73.40%	90.33%	94.06%	96.77%	98.81%
6Ks	baseline	66.36%	89.25%	91.77%	93.00%	93.68%
6Ks	+collocations	**68.56%**	90.50%	92.83%	93.97%	94.62%
6Ks	+clustering	66.71%	**91.77%**	**93.87%**	**95.38%**	**95.38%**
6Ks	+collocations+clustering	68.34%	90.02%	92.43%	93.53%	94.23%
5Kw	baseline	82.79%	93.07%	95.04%	96.48%	98.71%
5Kw	+collocations	82.84%	**93.62%**	**95.55%**	96.69%	**98.98%**
5Kw	+clustering	**82.88%**	93.54%	95.49%	**96.72%**	98.86%
5Kw	+collocations+clustering	**82.88%**	93.20%	95.27%	96.57%	98.92%

Table 3: The accuracies of appending collocations and K-means clustering. Here, w = word, s = sentence.

Word1	Word2	Kana1	Kana2
会社/company	情報/information	ガイシャ	ジョウホウ
人気/popular	商品/products	ヒトケ	ショウヒン
沖/oki, place name	地震/earthquake	オキ	ジシン
不動産/estate	会社/company	フドウサン	ガイシャ
半角/halfwidth	英数字/english letters, numbers	ハンカク	エイスウジ
生命/life	保険/ insurance	イノチ	ホケン

Figure 3: Examples of bi-gram collocations.

- first screen "precision", i.e., if the reference Kanji string is included in the first screen of the output list. Currently, we set the first screen includes top-9 candidates;
- "recall", i.e., if the reference Kanji string is included in the output list of the IME system.

By using the "precision" criteria, we hope to measure the goodness of the ranking of the output candidate list. The best situation is that, all the reference Kanji strings appear as the 1-best output of the IME system. Also, by using the "recall" criteria, we hope to measure the rate of missing Kanji candidates for the given input Kana sequences. A lot of Japanese family names or given names occur rarely even in the 2.5TB training data. However, we have to make sure they are included in the candidate list, since this significantly influences users' experience.

4.2 Appending Collections

We mined 423,617 bigram collocations by filtering the bigram set of the 2.5TB training data using Formula 3. Figure 3 shows several examples of collocations mined. Each entry in the collocation contains Kanji-style word sequence, POS tags, and Kana pronunciations. These entries are appended to our foundational Kana-Kanji conversion lexicon.

Table 3 shows the top-1/3/5 precisions, first screen precisions, and recalls achieved by the baseline and the baseline appended with collocations. From this table, we observe that in both test sets, the collocation appending approach achieved a better precision/recall than the baseline system. Also,

Input Kana	Reference Kanji	Our IME system (1-best)	Meaning
うさぎ	兎	うさぎ	rabbit
うけとり	受取	受け取り	receive
いりぐち	入り口	入口	entrance
いらいら	いらいら	イライラ	boredom
いのちをかける	命を懸ける	命をかける	risk one's life
いただく	頂く	いただく	accept/let's eat
いたす	致す	いたす	do
あなた	貴方	あなた	you
あさごはん	朝御飯	朝ごはん	breakfast
あげる	上げる	あげる	increase

Table 4: 10 examples in our test sets that reflects both the reference Kanji and the top-1 output of our IME system are acceptable results.

note that the first screen precisions already achieved more than 93% and the recalls achieved more than 93%. Through these accuracies, we believe that our current IME system has achieved an inspiring accuracy under these test sets.

Another interesting observation is that, there is a big jump (more than 10%) of the conversion accuracies from top-1 to top-3 precisions. We made an inside analysis of the cases that took the reference Kanji into the second or third positions of the final output. An important fact is that, a lot of entries in the test sets do take multiple forms (i.e., either Kana or Kanji sequences) as their appearances. In most cases, there are more than one candidates in the top-3 lists are acceptable to the real users. Table 4 lists 10 cases that take both the reference and the top-1 output of our IME system as acceptable results to the Kana inputs. Indeed, since the ranking of the candidates are mainly based on their frequencies in the 2.5TB training data, we believe the top-1 outputs generated by our IME system are more frequently used by most users and are more "commonly" used as habits. Thus, we believe the top-1 precision is a "conservative" estimation of the final precision and it is reasonable for use to also refer to the top-3/5 and first screen precisions to evaluate the improvements of our proposed approach.

Finally, the improvements on the 6Ks sentence level test set are much better than that achieved in the 23Kw and 5Kw word level sets. This reflects that appending of collections is more suitable for large-context input experience.

After appending bi-gram collocations, we also performed extracting collocations from Japanese single Bensetsu and several Bensetsus. The new accuracy (especially, Top-1 precision) in the sentence level test set was significantly better than current results. We wish to report the detailed techniques and results in our future publications.

4.3 K-means Clustering

By performing K-means clustering to the top-10 word classes, we finally obtained 2,811 word classes. After obtaining the new word class set, we retrained $P(c_i|c_{i-1})$ and $P(w_i|c_i)$ using the 2.5TB Web data.

The precisions and recalls by applying clustering to the baseline IME system are also shown in Table 3. From the table, we also obtained improvements on both precisions and recall. Also, the

improvements on sentence level test set with richer context information are better than that achieved in the word level test sets.

We finally combined our two proposals together, i.e, modify the original n-pos model by both appending collocations and sub-categorizing of POS tags. However, as shown in Table 3, the final results did not show a further better result than either of the single approaches. The main reason is that, the word classes for the collocations were based on the POS tags before sub-categorizing. This makes the collocations not sensitive to the changes of fin-grained POS tags. One solution to this problem is to enlarge the POS tags in the Japanese POS tagger, i.e., replacing the original IPA-style POS tags with our fine-grained POS tags. Since we do not have a training set with fine-grained POS tags, we wish to make use of the Expectation-Maximization algorithm (Dempster et al., 1977) to solve this problem by taking the fine-grained POS tag set as latent variable. Similar idea has been implemented for PCFG parsing with latent variables (Matsuzaki et al., 2005). We take this as a future work.

5 Conclusion

We have described two ways to improve current n-pos model for Japanese Kana-Kanji conversion. One way was to append the high-frequency collocations and the other way was to sub-categorize the huge POS sets. Experiments on large-scale data verified our proposals. Our Japanese IME system that implemented these ideas is completely free and has been used by millions of users running both on Windows-style PCs and Android-style smart phones. Future work includes enrich the feature set for unsupervised clustering, such as using the statistics, especially the context information[13], from the large-scale training data.

Acknowledgments

We thanks the anonymous reviewers for their comments and suggestions for improving the earlier version of this paper.

References

Dean, J. and Ghemawat, S. (2004). Mapreduce: simplified data processing on large clusters. In *Proceedings of OSDI*.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39:1–38.

Huang, L. and Chiang, D. (2005). Better k-best parsing. In *Proceedings of IWPT*.

Kneser, R. and Ney, H. (1993). Improved clustering techniques for class-based statistical language modeling. In *In Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*.

Komachi, M., Mori, S., and Tokunaga, H. (2008). Japanese, the ambiguous, and input methods (in japanese). In *Proceedings of the Summer Programming Symposium of Information Processing Society of Japan*.

Kudo, T., Komatsu, T., Hanaoka, T., Mukai, J., and Tabata, Y. (2011). Mozc: A statistical kana-kanji conversion system (in japanese). In *Proceedings of Japan Natural Language Processing*, pages 948–951.

[13] One reviewer also pointed out this, we express our thankfulness here.

Kudo, T., Yamamoto, K., and Matsumoto, Y. (2004). Applying conditional random fields to japanese morphological analysis. In Lin, D. and Wu, D., editors, *Proceedings of EMNLP 2004*, pages 230–237, Barcelona, Spain. Association for Computational Linguistics.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10:707–710.

Matsuzaki, T., Miyao, Y., and Tsujii, J. (2005). Probabilistic CFG with latent annotations. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 75–82, Ann Arbor, Michigan. Association for Computational Linguistics.

Mori, S., Nishimura, M., and Itoh, N. Word clustering for a word bi-gram model. In *Proceedings of ICSLP 1998*.

Mori, S., Tsuchiya, M., Yamaji, O., and Nagao, M. (1999). Kana-kanji conversion by a stochastic model (in japanese). *Journal of Information Processing Society of Japan*, 40(7).

Viterbi, A. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.