Proceedings of the Conference on Empirical Methods in Natural
Language Processing (EMNLP), Philadelphia, July 2002, pp. 248-256.
Association for Computational Linguistics.

Exploiting Headword Dependency and Predictive Clustering for Language Modeling

Jianfeng Gao
Microsoft Research, Asia
Beijing, 100080, China
jfgao@microsoft.com

Hisami Suzuki
Microsoft Research
Redmond WA 98052, USA
hisamis@microsoft.com

Yang Wen[*]
Department of Computer &
Information Sciences of
Tsinghua University, China

Abstract

This paper presents several practical ways of incorporating linguistic structure into language models. A headword detector is first applied to detect the headword of each phrase in a sentence. A permuted headword trigram model (PHTM) is then generated from the annotated corpus. Finally, PHTM is extended to a cluster PHTM (C-PHTM) by defining clusters for similar words in the corpus. We evaluated the proposed models on the realistic application of Japanese Kana-Kanji conversion. Experiments show that C-PHTM achieves 15% error rate reduction over the word trigram model. This demonstrates that the use of simple methods such as the headword trigram and predictive clustering can effectively capture long distance word dependency, and substantially outperform a word trigram model.

1 Introduction

In spite of its deficiencies, trigram-based language modeling still dominates the statistical language modeling community, and is widely applied to tasks such as speech recognition and Asian language text input (Jelinek, 1990; Gao *et al*., 2002).

Word trigram models are deficient because they can only capture local dependency relations, taking no advantage of richer linguistic structure. Many proposals have been made that try to incorporate linguistic structure into language models (LMs), but little improvement has been achieved so far in realistic applications because (1) capturing longer distance word dependency leads to higher-order *n*-gram models, where the number of parameters is usually too large to estimate; (2) capturing deeper linguistic relations in a LM requires a large amount of annotated training corpus and a decoder that assigns linguistic structure, which are not always available.

This paper presents several practical ways of incorporating long distance word dependency and linguistic structure into LMs. A headword detector is first applied to detect the headwords in each phrase in a sentence. A permuted headword trigram model (PHTM) is then generated from the annotated corpus. Finally, PHTM is extended to a cluster model (C-PHTM), which clusters similar words in the corpus.

Our models are motivated by three assumptions about language: (1) Headwords depend on previous headwords, as well as immediately preceding words; (2) The order of headwords in a sentence can freely change in some cases; and (3) Word clusters help us make a more accurate estimate of the probability of word strings. We evaluated the proposed models on the realistic application of Japanese Kana-Kanji conversion, which converts phonetic Kana strings into proper Japanese orthography. Results show that C-PHTM achieves a 15% error rate reduction over the word trigram model. This demonstrates that the use of simple methods can effectively capture long distance word dependency, and substantially outperform the word trigram model. Although the techniques in this paper are described in the context of Japanese Kana-Kanji conversion, we believe that they can be extended to other languages and applications.

This paper is organized as follows. Sections 2 and 3 describe the techniques of using headword

[*] This work was done while the author was visiting Microsoft Research Asia.

dependency and clustering for language modeling. Section 4 reviews related work. Section 5 introduces the evaluation methodology, and Section 6 presents the results of our main experiments. Section 7 concludes our discussion.

2 Using Headwords

2.1 Motivation

Japanese linguists have traditionally distinguished two types of words[1], content words (*jiritsugo*) and function words (*fuzokugo*), along with the notion of the *bunsetsu* (phrase). Each bunsetsu typically consists of one content word, called a *headword* in this paper, and several function words. Figure 1 shows a Japanese example sentence and its English translation[2].

[治療+に][専念+して][全快+まで][十分+な][療養+に][努め+る]

[*chiryou*+*ni*] [*sennen*+*shite*] [*zenkai*+*made*]
[treatment+to][concentration+do][full-recovery+until]

[*juubun*+*na*] [*ryouyou*+*ni*] [*tsutome*+*ru*]
[enough+ADN] [rest+for] [try+PRES]

'(One) concentrates on the treatment and tries to rest enough until full recovery'

Figure 1. A Japanese example sentence with bunsetsu and headword tags

In Figure 1, we find that some headwords in the sentence are expected to have a stronger dependency relation with their preceding headwords than with their immediately preceding function words. For example, the three headwords 治療~専念~全快 (*chiryou* 'treatment' ~ *sennen* 'concentrate' ~ *zenkai* 'full recovery') form a trigram with very strong semantic dependency. Therefore, we can hypothesize (in the trigram context) that headwords may be conditioned not only by the two immediately preceding words, but also by two previous headwords. This is our first assumption.

We also note that the order of headwords in a sentence is flexible in some sense. From the example in Figure 1, we find that if 治療~専念~全快 (*chiryou* 'treatment' ~ *sennen* 'concentrate' ~ *zenkai* 'full recovery') is a meaningful trigram, then its permutations (such as 全快~治療~専念 (*zenkai* 'full recovery' ~ *chiryou* 'treatment' ~ *sennen* 'concentrate')) should also be meaningful, because headword trigrams tend to capture an order-neutral semantic dependency. This reflects a characteristic of Japanese, in which arguments and modifiers of a predicate can freely change their word order, a phenomenon known as "scrambling" in linguistic literature. We can then introduce our second assumption: headwords in a trigram are permutable. Note that the permutation of headwords should be useful more generally beyond Japanese: for example, in English, *the book Mary bought* and *Mary bought a book* can be captured by the same headword trigram (*Mary* ~ *bought* ~ *book*) if we allow such permutations.

In this subsection, we have stated two assumptions about the structure of Japanese that can be exploited for language modeling. We now turn to discuss how to incorporate these assumptions in language modeling.

2.2 Permuted headword trigram model (PHTM)

A trigram model predicts the next word w_i by estimating the conditional probability $P(w_i|w_{i-2}w_{i-1})$, assuming that the next word depends only on two preceding words, w_{i-2} and w_{i-1}. The PHTM is a simple extension of the trigram model that incorporates the dependencies between headwords. If we assume that each word token can uniquely be classified as a headword or a function word, the PHTM can be considered as a cluster-based language model with two clusters, headword H and function word F. We can then define the conditional probability of w_i based on its history as the product of the two factors: the probability of the category (H or F), and the probability of w_i given its category. Let h_i or f_i be the actual headword or function word in a sentence, and let H_i or F_i be the category of the word w_i. The PHTM can then be formulated as:

$$P(w_i \mid \Phi(w_1 ... w_{i-1})) = \qquad (1)$$
$$P(H_i \mid \Phi(w_1 ... w_{i-1})) \times P(w_i \mid \Phi(w_1 ... w_{i-1}) H_i)$$
$$+ P(F_i \mid \Phi(w_1 ... w_{i-1})) \times P(w_i \mid \Phi(w_1 ... w_{i-1}) F_i)$$

where Φ is a function that maps the word history ($w_1 \ldots w_{i-1}$) onto equivalence classes.

[1] Or more correctly, morphemes. Strictly speaking, the LMs discussed in this paper are morpheme-based models rather than word-based, but we will not make this distinction in this paper.

[2] Square brackets demarcate the bunsetsu boundary, and + the morpheme boundary; the underlined words are the headwords. ADN indicates an adnominal marker, and PRES indicates a present tense marker.

$P(H_i|\Phi(w_1 \ldots w_{i-1}))$ and $P(F_i|\Phi(w_1 \ldots w_{i-1}))$ are category probabilities, and $P(w_i|\Phi(w_1 \ldots w_{i-1})F_i)$ is the word probability given that the category of w_i is function word. For these three probabilities, we used the standard trigram estimate (i.e., $\Phi(w_1 \ldots w_{i-1}) = (w_{i-2}w_{i-1})$). The estimation of headword probability is slightly more elaborate, reflecting the two assumptions described in Section 2.1:

$$\begin{aligned} P(w_i \mid \Phi(w_1 ... w_{i-1})H_i) = \lambda_1(&\lambda_2 P(w_i \mid h_{i-2}h_{i-1}H_i) \\ &+(1-\lambda_2)P(w_i \mid h_{i-1}h_{i-2}H_i)) \\ &+(1-\lambda_1)P(w_i \mid w_{i-2}w_{i-1}H_i)\,. \end{aligned} \quad (2)$$

This estimate is an interpolated probability of three probabilities: $P(w_i|h_{i-2}h_{i-1}H_i)$ and $P(w_i|h_{i-1}h_{i-2}H_i)$, which are the headword trigram probability with or without permutation, and $P(w_i|w_{i-2}w_{i-1}H_i)$, which is the probability of w_i given that it is a headword, where h_{i-1} and h_{i-2} denote the two preceding headwords, and λ_1, $\lambda_2 \in [0,1]$ are the interpolation weights optimized on held-out data.

The use of λ_1 in Equation (2) is motivated by the first assumption described in Section 2.1: headwords are conditioned not only on two immediately preceding words, but also on two previous headwords. In practice, we estimated the headword probability by interpolating the conditional probability based on two previous headwords $P(w_i|h_{i-2}h_{i-1}H_i)$ (and $P(w_i|h_{i-1}h_{i-2}H_i)$ with permutation), and the conditional probability based on two preceding words $P(w_i|w_{i-2}w_{i-1}H_i)$. If λ_1 is around zero, it indicates that this assumption does not hold in real data. Note that we did not estimate the conditional probability $P(w_i|w_{i-2}w_{i-1}h_{i-2}h_{i-1}H_i)$ directly, because this is in the form of a 5-gram, where the number of parameters are too large to estimate.

The use of λ_2 in Equation (2) comes from the second assumption in Section 2.1: headword trigrams are permutable. This assumption can be formulated as a co-occurrence model for headword prediction: that is, the probability of a headword is determined by the occurrence of other headwords within a window. However, in our experiments, we instead used an interpolated probability $\lambda_2 \times P(w_i|h_{i-2}h_{i-1}H_i) + (1-\lambda_2) \times P(w_i|h_{i-1}h_{i-2}H_i)$ for two reasons. First, co-occurrence models do not predict words from left to right, and are thus very difficult to interpolate with trigram models for decoding. Second, if we see *n*-gram models as one extreme that predicts the next word based on a strictly ordered word sequence, co-occurrence models go to the other extreme of predicting the next word based on a bag of previous words without taking word order into account at all. We prefer models that lie somewhere between the two extremes, and consider word order in a more flexible way. In PHTM of Equation (2), λ_2 represents the impact of word order on headword prediction. When $\lambda_2 = 1$ (i.e., the resulting model is a non-permuted headword trigram model, referred to as HTM), it indicates that the second assumption does not hold in real data. When λ_2 is around 0.5, it indicates that a headword bag model is sufficient.

2.3 Model parameter estimation

Assume that all conditional probabilities in Equation (1) are estimated using maximum likelihood estimation (MLE). Then

$$P(w_i \mid w_{i-2}w_{i-1}) = \begin{cases} P(H_i \mid w_{i-2}w_{i-1})P(w_i \mid w_{i-2}w_{i-1}H_i)\,, & w_i\text{: headword} \\ P(F_i \mid w_{i-2}w_{i-1})P(w_i \mid w_{i-2}w_{i-1}F_i)\,, & w_i\text{: function word} \end{cases}$$

is a strict equality when each word token is uniquely classified as a headword or a function word. This can be trivially proven as follows. Let C_i represent the category of w_i (H_i or F_i in our case). We have

$$\begin{aligned} &P(C_i \mid w_{i-2}w_{i-1}) \times P(w_i \mid w_{i-2}w_{i-1}C_i) \\ &= \frac{P(w_{i-2}w_{i-i}C_i)}{P(w_{i-2}w_{i-1})} \times \frac{P(w_{i-2}w_{i-1}C_i\, w_i)}{P(w_{i-2}w_{i-1}C_i)} \\ &= \frac{P(w_{i-2}w_{i-1}C_i\, w_i)}{P(w_{i-2}w_{i-1})} \end{aligned} \quad (3)$$

Since each word is uniquely assigned to a category, $P(C_i|w_i)$=1, and thus it follows that

$$\begin{aligned} P(w_{i-2}w_{i-1}C_i\, w_i) &= P(w_{i-2}w_{i-1}w_i) \times P(C_i \mid w_{i-2}w_{i-1}w_i) \\ &= P(w_{i-2}w_{i-1}w_i) \times P(C_i \mid w_i) \\ &= P(w_{i-2}w_{i-1}w_i)\,. \end{aligned} \quad (4)$$

Substituting Equation (4) into Equation (3), we get

$$\begin{aligned} &P(C_i \mid w_{i-2}w_{i-1}) \times P(w_i \mid w_{i-2}w_{i-1}C_i) \\ &= \frac{P(w_{i-2}w_{i-1}w_i)}{P(w_{i-2}w_{i-1})} = P(w_i \mid w_{i-2}w_{i-1})\,. \end{aligned} \quad (5)$$

Now, by separating the estimates of probabilities of headwords and function words, Equation (1) can be rewritten as:

$$P(w_i|\Phi(w_1 \ldots w_{i-1}))= \tag{6}$$
$$\begin{cases} \lambda_1(P(H_i \mid w_{i-2}w_{i-1})(\lambda_2 P(w_i \mid h_{i-2}h_{i-1}) \\ +(1-\lambda_2)P(w_i \mid h_{i-1}h_{i-2})) \\ +(1-\lambda_1)P(w_i \mid w_{i-2}w_{i-1}) \\ \qquad w_i\text{: headword} \\ P(w_i \mid w_{i-2}w_{i-1}) \\ \qquad w_i\text{: function word} \end{cases}$$

There are three probabilities to be estimated in Equation (6): word trigram probability $P(w_i|w_{i-2}w_{i-1})$, headword trigram probability $P(w_i|h_{i-2}h_{i-1})$ and $P(w_i|h_{i-1}h_{i-2})$ (where w_i is a headword), and category probability $P(H_i|w_{i-2}w_{i-1})$.

In order to deal with the data sparseness problem of MLE, we used a backoff scheme (Katz, 1987) for the parameter estimation. The backoff scheme recursively estimates the probability of an unseen *n*-gram by utilizing (*n*–1)-gram estimates. To keep the model size manageable, we also removed all *n*-grams with frequency less than 2.

In order to classify a word uniquely as *H* or *F*, we needed a mapping table where each word in the lexicon corresponds to a category. The table was generated in the following manner. We first assumed that the mapping from part-of-speech (POS) to word category is fixed. The tag set we used included 1,187 POS tags, of which 102 count as headwords in our experiments. We then used a POS-tagger to generate a POS-tagged corpus, from which we generated the mapping table[3]. If a word could be mapped to both *H* and *F*, we chose the more frequent category in the corpus. Using this mapping table, we achieved a 98.5% accuracy of headword detection on the test data we used.

Through our experiments, we found that $P(H_i|w_{i-2}w_{i-1})$ is a poor estimator of category probability; in fact, the unigram estimate $P(H_i)$ achieved better results in our experiments as shown in Section 6.1. Therefore, we also used the unigram estimate for word category probability in our experiments. The alternative model that uses the unigram estimate is given below:

$$P(w_i|\Phi(w_1 \ldots w_{i-1}))= \tag{7}$$
$$\begin{cases} \lambda_1(P(H_i)(\lambda_2 P(w_i \mid h_{i-2}h_{i-1}) \\ +(1-\lambda_2)P(w_i \mid h_{i-1}h_{i-2})) \\ +(1-\lambda_1)P(w_i \mid w_{i-2}w_{i-1}) \\ \qquad w_i\text{: headword} \\ P(w_i \mid w_{i-2}w_{i-1}) \\ \qquad w_i\text{: function word} \end{cases}$$

We will denote the models using trigram for category probability estimation of Equation (6) as T-PHTM, and the models using unigram for category probability estimation of Equation (7) as U-PHTM.

3 Using Clusters

3.1 Principle

Clustering techniques attempt to make use of similarities between words to produce a better estimate of the probability of word strings (Goodman, 2001).

We have mentioned in Section 2.2 that the headword trigram model can be thought of as a cluster-based model with two clusters, the headword and the function word. In this section, we describe a method of clustering automatically similar words and headwords. We followed the techniques described in Goodman (2001) and Gao *et al.* (2001), and performed experiments using predictive clustering along with headword trigram models.

3.2 Predictive clustering model

Consider a trigram probability $P(w_3|w_1w_2)$, where w_3 is the word to be predicted, called the *predicted word*, and w_1 and w_2 are context words used to predict w_3, called the *conditional words*. Gao *et al.* (2001) presents a thorough comparative study on various clustering models for Asian languages, concluding that a model that uses clusters for predicted words, called the *predictive clustering model*, performed the best in most cases.

Let $\overline{w_i}$ be the cluster which word w_i belongs to. In this study, we performed word clustering for words and headwords separately. As a result, we

[3] Since the POS-tagger does not identify phrases, our implementation does not identify precisely one headword for a phrase, but identify multiple headwords in the case of compounds.

have the following two predictive clustering models, (8) for words and (9) for headwords:

$$P(w_i \mid w_{i-2}w_{i-1}) = P(\overline{w_i} \mid w_{i-2}w_{i-1}) \times P(w_i \mid w_{i-2}w_{i-1}\overline{w_i}) \quad (8)$$

$$P(w_i \mid h_{i-2}h_{i-1}) = P(\overline{w_i} \mid h_{i-2}h_{i-1}) \times P(w_i \mid h_{i-2}h_{i-1}\overline{w_i}) \quad (9)$$

w_i: headword

Substituting Equations (8) and (9) into Equation (7), we get the cluster-based PHTM of Equation (10), referred to as C-PHTM.

$$P(w_i|\Phi(w_1 \ldots w_{i-1}))= \quad (10)$$
$$\begin{cases} \lambda_1(P(H_i)(\lambda_2 P(\overline{w_i} \mid h_{i-2}h_{i-1}) \times P(w_i \mid h_{i-2}h_{i-1}\overline{w_i}) \\ +(1-\lambda_2)P(\overline{w_i} \mid h_{i-1}h_{i-2}) \times P(w_i \mid h_{i-1}h_{i-2}\overline{w_i})) \\ +(1-\lambda_1)P(\overline{w_i} \mid w_{i-2}w_{i-1}) \times P(w_i \mid w_{i-2}w_{i-1}\overline{w_i}) \\ \qquad w_i\text{: headword} \\ \\ P(\overline{w_i} \mid w_{i-2}w_{i-1}) \times P(w_i \mid w_{i-2}w_{i-1}\overline{w_i}) \\ \qquad w_i\text{: function word} \end{cases}$$

3.3 Finding clusters: model estimation

In constructing clustering models, two factors were considered: how to find optimal clusters, and the optimal number of clusters.

The clusters were found automatically by attempting to minimize perplexity (Brown *et al.*, 1992). In particular, for predictive clustering models, we tried to minimize the perplexity of the training data of $P(\overline{w_i} \mid w_{i-1}) \times P(w_i \mid \overline{w_i})$. Letting *N* be the size of the training data, we have

$$\prod_{i=1}^{N} P(\overline{w_i} \mid w_{i-1}) \times P(w_i \mid \overline{w_i})$$
$$= \prod_{i=1}^{N} \frac{P(w_{i-1}\overline{w_i})}{P(w_{i-1})} \times \frac{P(\overline{w_i}w_i)}{P(W_i)}$$
$$= \prod_{i=1}^{N} \frac{P(\overline{w_i}w_i)}{P(w_{i-1})} \times \frac{P(w_{i-1}\overline{w_i})}{P(\overline{w_i})}$$
$$= \prod_{i=1}^{N} \frac{P(w_i)}{P(w_{i-1})} \times P(w_{i-1} \mid \overline{w_i})$$

Now, $\frac{P(w_i)}{P(w_{i-1})}$ is independent of the clustering used. Therefore, in order to select the best clusters, it is sufficient to try to maximize $\prod_{i=1}^{N} P(w_{i-1} \mid \overline{w_i})$.

The clustering technique we used creates a binary branching tree with words at the leaves. By cutting the tree at a certain level, it is possible to achieve a wide variety of different numbers of clusters. For instance, if the tree is cut after the sixth level, there will be roughly 2^6=64 clusters. In our experiments, we always tried the numbers of clusters that are the powers of 2. This seems to produce numbers of clusters that are close enough to optimal. In Equation (10), the optimal number of clusters we used was 2^7.

4 Relation to Previous Work

Our LMs are similar to a number of existing ones. One such model was proposed by ATR (Isotani and Matsunaga, 1994), which we will refer to as ATR model below. In ATR model, the probability of each word in a sentence is determined by the preceding content and function word pair. Isotani and Matsunaga (1994) reported slightly better results over word bigram models for Japanese speech recognition. Geutner (1996) interpolated the ATR model with word-based trigram models, and reported very limited improvements over word trigram models for German speech recognition.

One significant difference between the ATR model and our own lies in the use of predictive clustering. Another difference is that our models use separate probability estimates for headwords and function words, as shown in Equations (6) and (7). In contrast, ATR models are conceptually more similar to skipping models (Rosenfeld, 1994; Ney *et al.*, 1994; Siu and Ostendorf, 2000), where only one probability estimate is applied for both content and function words, and the word categories are used only for the sake of finding the content and function word pairs in the context.

Another model similar to ours is Jelinek (1990), where the headwords of the two phrases immediately preceding the word as well as the last two words were used to compute a word probability. The resulting model is similar to a 5-gram model. A sophisticated interpolation formula had to be used since the number of parameters is too large for direct estimation. Our models are easier to learn because they use trigrams. They also differ from Jelinek's model in that they separately estimate the probability for headwords and function words.

A significant number of sophisticated techniques for language modeling have recently been proposed in order to capture more linguistic structure from a larger context. Unfortunately, most of them suffer from either high computational cost or difficulty in obtaining enough manually parsed corpora for parameter estimation, which make it difficult to apply them successfully to realistic applications. For example, maximum entropy (ME) models (Rosenfeld, 1994) provide a nice framework for incorporating arbitrary knowledge sources, but training and using ME models is computationally extremely expensive.

Another interesting idea that exploits the use of linguistic structure is structured language modeling (SLM, Chelba and Jelinek, 2000). SLM uses a statistical parser trained on an annotated corpus in order to identify the headword of each constituent, which are then used as conditioning words in the trigram context. Though SLMs have been shown to significantly improve the performance of the LM measured in perplexity, they also pose practical problems. First, the performance of SLM is contingent on the amount and quality of syntactically annotated training data, but such data may not always be available. Second, SLMs are very time-intensive, both in their training and use.

Charniak (2001) and Roark (2001) also present language models based on syntactic dependency structure, which use lexicalized PCFGs that sum over the derivation probabilities. They both report improvements in perplexity over Chelba and Jelinek (2000) on the Wall Street Journal section of the Penn Treebank data, suggesting that syntactic structure can be further exploited for language modeling. The kind of linguistic structure used in our models is significantly more modest than that provided by parser-based models, yet offers practical benefits for realistic applications, as shown in the next section.

5 Evaluation Methodology

The most common metric for evaluating a language model is perplexity. Perplexity can be roughly interpreted as the expected branching factor of the test document when presented to a language model. Perplexity is widely used due to its simplicity and efficiency. However, the ultimate quality of a language model must be measured by its effect on the specific task to which it is applied, such as speech recognition. Lower perplexities usually result in lower error rates, but there are numerous counterexamples to this in the literature.

In this study, we evaluated our language models on the application of Japanese Kana-Kanji conversion, which is the standard method of inputting Japanese text by converting the text of syllabary-based Kana string into the appropriate combination of ideographic Kanji and Kana. This is a similar problem to speech recognition, except that it does not include acoustic ambiguity. Performance on this task is generally measured in terms of the character error rate (CER), which is the number of characters wrongly converted from the phonetic string divided by the number of characters in the correct transcript. The role of the language model is to select the word string (in a combination of Kanji and Kana) with the highest probability among the candidate strings that match the typed phonetic (Kana) string. Current products make about 5-10% errors in conversion of real data in a wide variety of domains.

For our experiments, we used two newspaper corpora: Nikkei and Yomiuri Newspapers. Both corpora have been word-segmented. We built language models from a 36-million-word subset of the Nikkei Newspaper corpus. We performed parameter optimization on a 100,000-word subset of the Yomiuri Newspaper (held-out data). We tested our models on another 100,000-word subset of the Yomiuri Newspaper corpus. The lexicon we used contains 167,107 entries.

In our experiments, we used the so-called "N-best rescoring" method. In this method, a list of hypotheses is generated by the baseline language model (a word trigram model in this study[4]), which is then rescored using a more sophisticated LM. Due to the limited number of hypotheses in the N-best list, the second pass may be constrained by the first pass. In this study, we used the 100-best list. The "oracle" CER (i.e., the CER among the hypotheses with the minimum number of errors) is presented in Table 1. This is the upper bound on performance in our experiments. The performance of the conversion using the baseline trigram model is much better than the state-of-the-art performance currently available in the marketplace. This may be due to the large amount of training data we used, and to the similarity between the training and the test data. We also notice that the "oracle" CER is

[4] For the detailed description of the baseline trigram model, see Gao *et al*. (2002).

relatively high due to the high out-of-vocabulary rate, which is 1.14%. Because we have only limited room for improvement, the reported results of our experiments in this study may be underestimated.

Baseline Trigram	Oracle of 100-best
3.73%	1.51%

Table 1. CER results of baseline and 100-best list

6 Results and Discussion

6.1 Impact of headword dependency and predictive clustering

We applied a series of language models proposed in this paper to the Japanese Kana-Kanji conversion task in order to test the effectiveness of our techniques. The results are shown in Table 2. The baseline result was obtained by using a conventional word trigram model. HTM stands for the headword trigram model of Equation (6) and (7) without permutation (i.e., λ_2=1), while PHTM is the model with permutation. The T- and U-prefixes refer to the models using trigram (Equation (6)) or unigram (Equation (7)) estimate for word category probability. The C-prefix, as in C-PHTM, refers to PHTM with predictive clustering (Equation (10)). For comparison, we also include in Table 2 the results of using the predictive clustering model without taking word category into account, referred to as *predictive clustering trigram model* (PCTM). In PCTM, the probability for all words is estimated by $P(\overline{w_i} \mid w_{i-2}w_{i-1}) \times P(w_i \mid w_{i-2}w_{i-1}\overline{w_i})$.

Model	λ_1	λ_2	CER	CER reduction
Baseline	----	----	3.73%	----
T-HTM	0.2	1	3.54%	5.1%
U-HTM	0.2	1	3.41%	8.6%
T-PTHM	0.2	0.7	3.53%	5.4%
U-PHTM	0.2	0.7	3.34%	10.5%
PCTM	----	----	3.44%	7.8%
C-HTM	0.3	1	3.23%	13.4%
C-PHTM	0.3	0.7	3.17%	15.0%

Table 2. Comparison of CER results

In Table 2, we find that for both PHTM and HTM, models U-HTM and U-PHTM achieve better performance than models T-HTM and T-PHTM. Therefore, only models using unigram for category probability estimation are used for further experiments, including the models with predictive clustering.

By comparing U-HTM with the baseline model, we can see that the headword trigram contributes greatly to the CER reduction: U-HTM outperformed the baseline model by about 8.6% in error rate reduction. HTM with headword permutation (U-PHTM) achieves further improvements of 10.5% CER reduction against the baseline. The contribution of predictive clustering is also very encouraging. Using predictive clustering alone (PCTM), we reduced the error rate by 7.8%.

What is particularly noteworthy is that the combination of both techniques leads to even larger improvements: for both HTM and PHTM, predictive clustering (C-HTM and C-PHTM) brings consistent improvements over the models without clustering, achieving the CER reduction of 13.4% and 15.0% respectively against the baseline model, or 4.8% and 4.5% against the models without clustering.

In sum, considering the good performance of our baseline system and the upper bound on performance improvement due to the 100-best list as shown in Table 1, the improvements we obtained are very promising. These results demonstrate that the simple method of using headword trigrams and predictive clustering can be used to effectively improve the performance of word trigram models.

6.2 Comparsion with other models

In this subsection, we present a comparison of our models with some of the previously proposed models, including the higher-order *n*-gram models, skipping models, and the ATR models.

Higher-order *n*-gram models refer to those *n*-gram models in which *n*>3. Although most of the previous research showed little improvement, Goodman (2001) showed recently that, with a large amount of training data and sophisticated smoothing techniques, higher-order *n*-gram models could be superior to trigram models.

The headword trigram model proposed in this paper can be thought of as a variation of a higher order *n*-gram model, in that the headword trigrams capture longer distance dependencies than trigram models. In order to see how far the dependency goes within our headword trigram models, we plotted the distribution of headword trigrams (y-axis) against the *n* of the word *n*-gram were it to be captured by the word *n*-gram (x-axis) in Figure 2. For example, given a word sequence $w_1w_2w_3w_4w_5w_6$, and if w_1, w_3 and w_6 are headwords, then the headword trigram $P(w_6|w_3w_1)$ spans the same distance as the word 6-gram model.

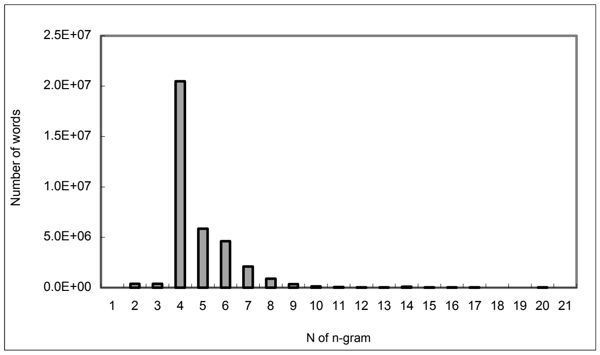

Figure 2. Distribution of headword trigrams against the *n* of word *n*-gram

From Figure 2, we can see that approximately 95% of the headword trigrams can be captured by the higher-order *n*-gram model with the value of *n* smaller than 7. Based on this observation, we built word *n*-gram models with the values of *n*=4, 5 and 6. For all *n*-gram models, we used the interpolated modified absolute discount smoothing method (Gao *et al*., 2001), which, in our experiments, achieved the best performance among the state-of-the-art smoothing techniques. Results showed that the performance of the higher-order word *n*-gram models becomes saturated quickly as *n* grows: the best performance was achieved by the word 5-gram model, with the CER of 3.71%. Following Goodman (2001), we suspect that the poor performance of these models is due to the data sparseness problem.

Skipping models are an extension of an *n*-gram model in that they predict words based on *n* conditioning words, except that these conditioning words may not be adjacent to the predicted word. For instance, instead of computing $P(w_i|w_{i-2}w_{i-1})$, a skipping model might compute $P(w_i|w_{i-3}w_{i-1})$ or $P(w_i|w_{i-4}w_{i-2})$. Goodman (2001) performed experiments of interpolating various kinds of higher-order *n*-gram skipping models, and obtained a very limited gain. Our results confirm his results and suggest that simply extending the context window by brute-force can achieve little improvement, while the use of even the most modest form of structural information such as the identification of headwords and automatic clustering can help improve the performance.

We also compared our models with the trigram version of the ATR models discussed in Section 4, in which the probability of a word is conditioned by the preceding content and function word pair. We performed experiments using the ATR models as described in Isotani and Matsunaga (1994). The results show that the CER of the ATR model alone is much higher than that of the baseline model, but when interpolated with a word trigram model, the CER is slightly reduced by 1.6% from 3.73% to 3.67%. These results are consistent with those reported in previous work. The difference between the ATR model and our models indicates that the predictions of headwords and function words can better be done separately, as they play different semantic and syntactic roles capturing different dependency structure.

6.3 Discussion

In order to better understand the effect of the headword trigram, we have manually inspected the actual improvements given by PHTM. As expected, many of the improvements seem to be due to the use of larger context: for example, the headword trigram 消費～支出～減少 (*shouhi* 'consume' ~ *shishutsu* 'expense' ~ *genshou* 'decrease') contributed to the correct conversion of the phonetic string げんしょう *genshou* into 減少 *genshou* 'decrease' rather than 現象 *genshou* 'phenomenon' in the context of 消費支出初めての<u>減少</u> *shouhi shishutsu hajimete no genshou* 'consumer spending decreases for the first time'.

On the other hand, the use of headword trigrams and predictive clustering is not without side effects. The overall gain in CER was 15% as we have seen above, but a closer inspection of the conversion results reveals that while C-PHTM corrected the conversion errors of the baseline model in 389 sentences (8%), it also introduced new conversion errors in 201 sentences (4.1%). Among the newly introduced errors, one type of error is particularly worth noting: these are the errors where the candidate conversion preferred by the HTM is grammatically impossible or unlikely. For example, 米国に侵攻できる (*beikoku-ni shinkou-dekiru*, USA-to invade-can 'can invade USA') was misconverted as 米国に新興できる (*beikoku-ni shinkou-dekiru*, USA-to new-can), even though 侵攻 *shinkou* 'invade' is far more likely to be preceded by the morpheme に *ni* 'to', and 新興 *shinkou* 'new' practically does not precede できる *dekiru* 'can'. The HTM does not take these function words into account, leading to a grammatically impossible or implausible conversion. Finding the types of errors introduced by particular modeling assumptions in this manner and addressing them individually will be the next step for further improvements in the conversion task.

7 Conclusion

We proposed and evaluated a new language model, the permuted headword trigram model with clustering (C-PHTM). We have shown that the simple model that combines the predictive clustering with a headword detector can effectively capture structure in language. Experiments show that the proposed model achieves an encouraging 15% CER reduction over a conventional word trigram model in a Japanese Kana-Kanji conversion system. We also compared C-PTHM to several similar models, showing that our model has many practical advantages, and achieves substantially better performance.

One issue we did not address in this paper was the language model size: the models that use HTM are larger than the baseline model we compared the performance with. Though we did not pursue the issue of size reduction in this paper, there are many known techniques that effectively reduce the model size while minimizing the loss in performance. One area of future work is therefore to reduce the model size. Other areas include the application of the proposed model to a wider variety of test corpora and to related tasks.

Acknowledgements

We would like to thank Ciprian Chelba, Bill Dolan, Joshua Goodman, Changning Huang, Hang Li and Yoshiharu Sato for their comments on early thoughts and drafts of the paper. We would also like to thank Hiroaki Kanokogi, Noriko Ishibashi and Miyuki Seki for their help in our experiments.

References

Brown, Peter F., Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-Based N-gram Models of Natural Language. *Computational Linguistics*, 18-4: 467-479.

Charniak, Eugene. 2001. Immediate-head parsing for language models. In *ACL/EACL 2001*, pp.124-131.

Chelba, Ciprian and Frederick Jelinek. 2000. Structured Language Modeling. *Computer Speech and Language*, Vol. 14, No. 4. pp 283-332.

Gao, Jianfeng, Joshua T. Goodman and Jiangbo Miao. 2001. The use of clustering techniques for language model – application to Asian language. *Computational Linguistics and Chinese Language Processing*. Vol. 6, No. 1, pp 27-60.

Gao, Jianfeng, Joshua Goodman, Mingjing Li and Kai-Fu Lee. 2002. Toward a unified approach to statistical language modeling for Chinese. *ACM Transactions on Asian Language Information Processing*, Vol. 1, No. 1, pp 3-33.

Geutner, Petra. 1996. Introducing linguistic constraints into statistical language modeling. In *International Conference on Spoken Language Processing*, Philadelphia, USA. pp.402-405.

Goodman, Joshua T. 2001. A bit of progress in language modeling. *Computer Speech and Language.* October, 2001, pp 403-434.

Goodman, Joshua T., and Jianfeng Gao. 2000. Language model size reduction by pruning and clustering. *ICSLP-2000*, Beijing.

Isotani, Ryosuke and Shoichi Matsunaga. 1994. A stochastic language model for speech recognition integrating local and global constraints. *ICASSP-94*, pp. 5-8.

Jelinek, Frederick. 1990. Self-organized language modeling for speech recognition. In A. Waibel and K. F. Lee (eds.), *Readings in Speech Recognition*, Morgan-Kaufmann, San Mateo, CA. pp. 450-506.

Katz, S. M. 1987. Estimation of probabilities from sparse data for other language component of a speech recognizer. *IEEE transactions on Acoustics, Speech and Signal Processing*, 35(3): 400-401.

Ney, Hermann, Ute Essen and Reinhard Kneser. 1994. On structuring probabilistic dependences in stochastic language modeling. *Computer Speech and Language*, 8: 1-38.

Roark, Brian. 2001. Probabilistic top-down parsing and language modeling. *Computational Linguistics*, 17-2: 1-28.

Rosenfeld, Ronald. 1994. *Adaptive statistical language modeling: a maximum entropy approach*. Ph.D. thesis, Carnegie Mellon University.

Siu, Manhung and Mari Ostendorf. 2000. Variable n-grams and extensions for conversational speech language modeling. *IEEE Transactions on Speech and Audio Processing*, 8: 63-75.