Computational Linguistics and Chinese Language Processing
Vol. 6, No. 1, February 2001, pp. 27-60
© Computational Linguistics Society of R.O.C.

The Use of Clustering Techniques for Language Modeling – Application to Asian Language

Jianfeng Gao*, Joshua T. Goodman+, Jiangbo Miao**

Abstract

Cluster-based *n*-gram modeling is a variant of normal word-based *n*-gram modeling. It attempts to make use of the similarities between words. In this paper, we present an empirical study of clustering techniques for Asian language modeling. Clustering is used to improve the performance (i.e. perplexity) of language models as well as to compress language models. Experimental tests are presented for cluster-based trigram models on a Japanese newspaper corpus and on a Chinese heterogeneous corpus. While the majority of previous research on word clustering has focused on how to get the best clusters, we have concentrated our research on the best way to use the clusters. Experimental results show that some novel techniques we present work much better than previous methods, and achieve more than 40% size reduction at the same level of perplexity.

1. Introduction

Statistical language modelling (SLM) has been successfully applied in many domains, such as speech recognition, optical character recognition, machine translation, spelling correction, information retrieval, and spoken language understanding [Jelinek, 1990; Church, 1988; Brown *et al.*, 1990; Kernighan *et al.*, 1990; Miller *et al.*, 1999; Zue, 1995]. The dominant technology in SLM is *n*-gram models.

Typically, *n*-gram models are trained on very large corpora. In constructing *n*-gram models, we always face two problems. First, for a general domain model, large amounts of training data can lead to models that are too large for realistic applications. On the other hand, for specific domains, *n*-gram models usually suffer from the data sparseness problem because large amounts of domain-specific data are usually not available.

When *n*-gram models are used, we can define clusters for similar words in a corpus. We

* Microsoft Research, Asia, Beijing, 100080, P.R.C. E-mail: jfgao@microsoft.com
+ Microsoft Research, Redmond Washington 98052, USA. E-mail: joshuago@microsoft.com
** Department of Computer & Information Sciences University of Delaware, USA. This work was done while the author was visiting Microsoft Research Asia.

thus extend word-based *n*-gram models to cluster-based *n*-gram models. This has been demonstrated as an effective way to handle the data sparseness problem. Recent research also shows that cluster-based *n*-gram models are effective for rapid domain adaptation, training on small data sets, and reducing the memory requirements for realistic applications.

Extending our previous work in [Goodman, 2001; Gao *et al.*, 2001; Goodman and Gao, 2000], this paper presents an empirical study of clustering techniques for Asian language modeling. Clustering is used to improve the performance (i.e. perplexity) of language models as well as to compress language models. Experimental tests will be presented for cluster-based trigram models on a Japanese newspaper corpus of more than 10 million words, and on a Chinese heterogeneous corpus of more than 11 million characters. The majority of the previous research on word clustering has focused on how to get the best clusters. We have concentrated our research on the best way to use the clusters. Experimental results show that some novel techniques work much better than previous methods.

This paper is structured as follows: In the remainder of this section, we present an introduction to *n*-gram models, smoothing, and performance evaluation. In Section 2, we briefly review previous work on word clustering and cluster-based *n*-gram models. In Section 3, we present our technique of using clusters for trigram models. In Section 4, we describe our method for finding clusters. In Section 5, we present the results of our main experiments. Finally, we present our conclusions in Section 6.

1.1 *N*-gram models

The classic task of *language modeling* is to predict the next word given the previous words. The *n*-gram model is the usual approach. It states the task of predicting the next word as attempting to estimate the conditional probability:

$$P(w_n) = P(w_n \mid w_1 \cdots w_{n-1})\,. \tag{1}$$

In practice, the cases of *n*-gram models that people usually use are n=2,3,4, referred to as a *bigram*, a *trigram*, and a *four-gram* model, respectively. For example, in trigram models, the probability of a word is assumed to depend only on the two previous words:

$$P(w_n \mid w_1 \cdots w_{n-1}) \approx P(w_n \mid w_{n-2} w_{n-1})\,. \tag{2}$$

An estimate of the probability $P(w_i \mid w_{i-2} w_{i-1})$ is given by Equation (3), called the *maximum likelihood estimation* (MLE):

$$P(w_i \mid w_{i-2} w_{i-1}) = \frac{C(w_{i-2} w_{i-1} w_i)}{C(w_{i-2} w_{i-1})}\,, \tag{3}$$

where $C(w_{i-2} w_{i-1} w_i)$ represents the number of times the sequence $w_{i-2} w_{i-1} w_i$ occurs in the training text.

A difficulty with this approximation is that for word sequences that do not occur in the training text, where $C(w_{i-2}w_{i-1}w_i) = 0$, the predicted probability is 0. This makes it impossible for a system, such as a speech recognition system, to accept such a 0 probability sequence. Thus, these probabilities are typically smoothed [Chen and Goodman, 1999]: some probability is removed from all non-zero counts, and used to add probability to the 0 count items. The added probability is typically in proportion to some less specific, but less noisy model. For trigram models, typically, a formula of the following form is used:

$$P(w_i \mid w_{i-2}w_{i-1}) = \begin{cases} \dfrac{C(w_{i-2}w_{i-1}w_i) - D(C(w_{i-2}w_{i-1}w_i))}{C(w_{i-2}w_{i-1})} & \text{if } C(w_{i-2}w_{i-1}w_i) > 0 \\ \alpha(w_{i-2}w_{i-1})P(w_i \mid w_{i-1}) & \text{otherwise} \end{cases}, \quad (4)$$

where $\alpha(w_{i-2}w_{i-1})$ is a normalization factor, and is defined in such a way that the probabilities sum to 1. The function $D(C(w_{i-2}w_{i-1}w_i))$ is a discount function. It can, for instance, have a constant value, in which case the technique is called “Absolute Discounting”, or it can be a function estimated using the Good-Turing method, in which case the technique is called Good-Turing or Katz smoothing [Katz, 1987; Chen and Goodman 1999].

1.2 Performance evaluation

The most common metric for evaluating a language model is *perplexity*. Formally, the word perplexity PP_W of a model is the reciprocal of the geometric average probability assigned by the model to each word in the test set. It is defined as

$$PP_W = 2^{-\frac{1}{N_W}\sum_{i=1}^{N_W}\log_2 P(w_i|w_{i-2}w_{i-1})}, \quad (5)$$

where N_W is the total number of words in the test set. The perplexity can be roughly interpreted as the geometric mean of the branching factor of the test document when presented to the language model. Clearly, lower perplexities are better.

For applications, such as speech recognition, handwriting recognition, and spelling correction, it is generally assumed that lower perplexity correlates with better performance. In [Gao *et al*., 2001], we presented results that indicate this correlation is especially strong when the *n*-gram model is applied to the application of pinyin to Chinese character conversion, which is a similar problem to speech recognition.

2. Word Cluster and Cluster-based N-grams

For any given assignment of a word w_i to a cluster (also called a class) c_i, there may be many to many mappings; i.e. a word w_i may belong to more than one cluster, and a cluster c_i will typically contain more than one word. For the sake of simplicity, in this paper, we assume that a word w_i can only be uniquely mapped to its own cluster c_i, which is called hard clustering. The cluster-based *n*-gram model is a variant of the word-based *n*-gram model that uses the frequency of sequences of clusters to help produce a more knowledgeable estimate of the probability of word strings. The basic cluster-based *n*-gram model defines the conditional probability of a word w_i based on its history as the product of the two factors: the probability of the cluster given the preceding clusters, and the probability of a particular word given the cluster [Brown *et al.*, 1990]. For example, in cluster-based trigram models, we have

$$P(w_i \mid w_{i-2} w_{i-1}) = P(w_i \mid c_i) \times P(c_i \mid c_{i-2} c_{i-1}) . \tag{6}$$

The MLE of the probability of the word given the cluster, and the probability of the cluster given the two previous clusters can be computed as follows:

$$P(w_i \mid c_i) = \frac{C(w_i)}{C(c_i)} , \tag{7}$$

$$P(c_i \mid c_{i-2} c_{i-1}) = \frac{C(c_{i-2} c_{i-1} c_i)}{C(c_{i-2} c_{i-1})} . \tag{8}$$

A large amount of previous research has focused on how to best cluster similar words together. The proposed methods can be roughly grouped into two categories: (1) knowledge based clustering, and (2) data-driven clustering.

In knowledge based clustering, words are clustered based on the syntactic/semantic information we have for the language and the task [Jelinek, 1990; Heeman, 1999; Heeman and Allen, 1997; Placeway *et al.*, 1993; Issar and Ward, 1994; Ward and Young, 1993]. For example, part of speech (POS) tags can be generally used to produce a small number of clusters although this may lead to significantly increased perplexity [Srinivas, 1996; Niesler *et al.*, 1998]. Alternatively, if we have domain knowledge, it is often advantageous to cluster words that have a similar semantic functional role together. For example, [Issar and Ward, 1994] used tags like CITY and AIRLINE for an airline information system. There has also been some interesting research on word clustering for Chinese language. For example, [Yang *et al.*, 1994] present a method in which Chinese words are simply clustered according to their starting and ending characters. It assumes that because almost every Chinese character is a morpheme with its own meaning, very often words having the same starting or ending characters share some common linguistic properties and, thus, can form a word cluster. A good example is the cluster containing “yesterday” (昨天), “tomorrow” (明天), “everyday” (每天), “Sunday” (星期天) etc.

In data-driven clustering, words are clustered automatically in a such way that the overall perplexity of the corpus is minimized [Brown *et al.*, 1992]. A greedy search algorithm is generally used for clustering. It basically works as follows. First, each word is initialized to a random cluster. Then, at each iteration, every word is moved to a cluster such that the resulting model has the minimum perplexity. The algorithm converges when no single word can be moved to another cluster in a way that reduces the perplexity of the cluster-based *n*-gram model. Most previous research has found only small differences between different techniques for finding clusters [Kneser and Ney, 1993; Yamamoto and Sagisaka, 1999; Ueberla, 1996; Pereira *et al.*, 1993; Bellegarda *et al.*, 1996; Bai *et al.*, 1998]. One result, however, is that automatically derived clusters outperform POS tags [Niesler *et al.*, 1998], at least when there is enough training data [Ney *et al.*, 1994].

While cluster-based *n*-gram models often offer no perplexity reduction in comparison to word-based *n*-gram models, it is beneficial to smooth the word-based *n*-gram model via either backoff or interpolation methods (although the improvement is marginal) [Maltese and Mancini, 1992; Miller and Alleva, 1997]. One typical example is a combined model where the cluster-based *n*-gram model can be linearly interpolated with a normal word-based *n*-gram model [Brown *et al.*, 1992]:

$$\lambda P(w_i \mid w_{i-2} w_{i-1}) + (1-\lambda) P(w_i \mid c_i) \times P(c_i \mid c_{i-2} c_{i-1}) \tag{9}$$

where λ is the interpolation weight optimized on heldout data.

In this study, we focused our research on novel techniques for using clusters rather than different ways of finding clusters. We also noticed that all realistic applications have memory constraints. Therefore, we concentrated our experiments on finding the best way to use cluster-based *n*-gram models together with word-based *n*-gram models to seek the optimum balance between memory storage and perplexity. In Section 5, most of our experimental results will be presented in the form of size/perplexity curves.

3. Using Clusters

In this section, we will describe our techniques for using clusters, which are a bit different than traditional clustering as shown in Equation (6). As a typical example, consider the 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 word*. Either the predicted word or the conditional word can be clustered when building cluster-based trigram models. Therefore, there are three basic forms of cluster-based trigram models. When using clusters for the predicted word as shown in Equation (10), we get the first kind of cluster-based trigram model, called *predictive clustering*. When using clusters for the conditional word as shown in Equation (11), we get the second model, called *conditional clustering*. When using clusters for both the predicted word and the conditional word, we get Equation (12), called *combined clustering*:

$$P(w_i \mid w_{i-2}w_{i-1}) = P(c_i \mid w_{i-2}w_{i-1}) \times P(w_i \mid w_{i-2}w_{i-1}c_i)\,, \tag{10}$$

$$P(w_i \mid w_{i-2}w_{i-1}) = P(w_i \mid c_{i-2}c_{i-1})\,, \tag{11}$$

$$P(w_i \mid w_{i-2}w_{i-1}) = P(c_i \mid c_{i-2}c_{i-1}) \times P(w_i \mid c_{i-2}c_{i-1}c_i)\,. \tag{12}$$

In what follows, each technique will be discussed in detail, and illustrated by an example.

3.1 Predictive clustering

Consider a probability such as *P*(*Tuesday*| *party on*). Perhaps the training data contains no instances of the phrase "*party on Tuesday*", although other phrases such as "*party on Wednesday*" and "*party on Friday*" do appear. We can put words into clusters, such as the word "*Tuesday*" into the cluster *WEEKDAY*. Now, we can consider the probability of the word "*Tuesday*" given the phrase "*party on*", and also given that the next word is a *WEEKDAY*. We will denote this probability by *P*(*Tuesday* | *party on WEEKDAY*). We can then decompose the probability

$$P(Tuesday \mid party\ on) \ = \ P(WEEKDAY \mid party\ on) \times \mathrm{P}(Tuesday \mid party\ on\ WEEKDAY).$$

When each word belongs to only one cluster, this decomposition is a strict equality. This can be trivially proven as follows:

$$\begin{aligned} P(c_i \mid w_{i-2}w_{i-1}) \times P(w_i \mid w_{i-2}w_{i-1}c_i) &= \frac{P(w_{i-2}w_{i-1}c_i)}{P(w_{i-2}w_{i-1})} \times \frac{P(w_{i-2}w_{i-1}c_iw_i)}{P(w_{i-2}w_{i-1}c_i)} \\ &= \frac{P(w_{i-2}w_{i-1}c_iw_i)}{P(w_{i-2}w_{i-1})}\,. \end{aligned} \tag{13}$$

Now, since each word belongs to a single cluster, $P(c_i|w_i)$=1, it follows that

$$\begin{aligned} P(w_{i-2}w_{i-1}c_iw_i) &= P(w_{i-2}w_{i-1}w_i) \times P(c_i \mid w_{i-2}w_{i-1}w_i) \\ &= P(w_{i-2}w_{i-1}w_i) \times P(c_i \mid w_i) \\ &= P(w_{i-2}w_{i-1}w_i)\,. \end{aligned} \tag{14}$$

Substituting Equation (14) into Equation (13), we get

$$P(c_i \mid w_{i-2}w_{i-1}) \times P(w_i \mid w_{i-2}w_{i-1}c_i) = \frac{P(w_{i-2}w_{i-1}w_i)}{P(w_{i-2}w_{i-1})} = P(w_i \mid w_{i-2}w_{i-1})\,. \tag{15}$$

Now, although Equation (15) is a strict equality, when smoothing is taken into consideration, using the clustered probability will be more accurate than using the non-clustered probability. For instance, even if we have never seen an example of "*party on*

Tuesday", perhaps we have seen examples of other phrases, such as "*party on Wednesday*"; thus, the probability $P(WEEKDAY \mid party\ on)$ will be relatively high. Furthermore, although we may never have seen an example of "*party on WEEKDAY Tuesday*", after we backoff or interpolate with a lower order model, we may able to accurately estimate $P(Tuesday|on\ WEEKDAY)$. Thus, our smoothed clustered estimate may be a good one. We call this particular kind of clustering *predictive clustering*. The general form is Equation (10).

3.2 Conditional clustering

On the other hand, we can also cluster the words we are conditioning on. For instance, if "*party*" is in the cluster *EVENT* and "*on*" is in the cluster "*PREPOSITION*", then we can write

$$P(Tuesday \mid party\ on) \approx P(Tuesday \mid EVENT\ PREPOSITION).$$

We call this kind of clustering *conditional clustering*. The general form is Equation (11).

3.3 Combined clustering

It is also possible to combine both predictive and conditional clustering, and, in fact, for some applications, this combination works better than either one separately. Thus, we can compute

$$P(Tuesday \mid party\ on) = \\ P(WEEKDAY \mid EVENT\ PREPOSITION) \times \mathrm{P}(Tuesday \mid EVENT\ PREPOSITION\ WEEKDAY).$$

We call this kind of clustering *combined clustering*. The general form is Equation (12). Equation (12) is a generalization of predictive clustering of Equation (10), in which case we used no clustering for conditional words. Equation (12) is also a generalization of conditional clustering of Equation (11), in which case we used no clustering for predicted words. Also notice that the combined cluster-based trigram model of Equation (12) is actually a generalization of a technique invented at IBM (Brown *et al.*, 1992), which uses the approximation $P(w_i|c_{i-2}\ c_{i-1}\ c_i) \approx P(w_i|c_i)$ to get

$$P(Tuesday \mid party\ on) \approx \\ P(WEEKDAY| EVENT\ PREPOSITION) \times \mathrm{P}(Tuesday \mid WEEKDAY).$$

The approximation is suboptimal unless we use high (count) cutoffs for bigrams and trigrams. Given that combined clustering uses more information than regular IBM clustering, we assumed that it would lead to improvements. As will be shown in Section 5, it works about the same or a little better, at least when interpolated with a normal word-based trigram model.

4. Finding Clusters

As described in Section 2, a large number of techniques for finding clusters have been proposed, but previous studies showed that no one technique outperforms other significantly. In this study, we did not explore different techniques for finding clusters, but simply picked one we thought would be good, based on previous research.

There is no need for the clusters used for different positions to be the same. In particular, for a model like IBM clustering, with $P(w_i|c_i)\times P(c_i|c_{i-2}\ c_{i-1})$, we call the cluster c_i a predictive cluster, and the clusters c_{i-2} and c_{i-1} conditional clusters. The predictive and conditional clusters can be different [Yamamoto and Sagisaka, 1999]. For instance, consider a pair of words like "*a*" and "*an*". In general, "*a*" and "*an*" can follow the same words, and so, for predictive clustering, belong to the same cluster. However, there are very few words that can follow both "*a*" and "*an*", and so, for conditional clustering, they belong to different clusters. We have also found in experiments that the optimal numbers of clusters used for predictive and conditional clustering are different. In this paper, we always optimize both the number of conditional and predictive clusters separately, and reoptimize for each technique on each training data set. This is a very time consuming task, since each time the number of clusters is changed, the models must be rebuilt from scratch. We always try numbers of clusters that are powers of 2, e.g. 1, 2, 4, etc. This seems to produce numbers of clusters that are close enough to optimal.

The clusters are found automatically using a tool that attempts to minimize perplexity. In particular, for conditional clusters, we try to minimize the perplexity of the training data for a bigram of the form $P(w_i|c_{i-1})$, which is equivalent to maximizing

$$\prod_{i=1}^{N} P(w_i \mid c_{i-1})\,. \tag{16}$$

For predictive clusters, we try to minimize the perplexity of the training data of $P(c_i|w_{i-1})\times P(w_i|c_i)$. We do not minimize $P(c_i|w_{i-1})\times P(w_i|w_{i-1}\ c_i)$ because we are doing our minimization on unsmoothed training data, and the latter formula would, thus, be equal to $P(w_i|w_{i-1})$ for any clustering. If we were to use the method of leaving-one-out (Kneser and Ney, 1993), then we could use the latter formula, but the approach would be more difficult. Now,

$$\begin{aligned}\prod_{i=1}^{N} P(c_i \mid w_{i-1})\times P(w_i \mid c_i) &= \prod_{i=1}^{N} \frac{P(w_{i-1}c_i)}{P(w_{i-1})}\times\frac{P(c_i w_i)}{P(c_i)} \\ &= \prod_{i=1}^{N} \frac{P(c_i w_i)}{P(w_{i-1})}\times\frac{P(w_{i-1}c_i)}{P(c_i)} \\ &= \prod_{i=1}^{N} \frac{P(w_i)}{P(w_{i-1})}\times P(w_{i-1} \mid c_i)\,.\end{aligned} \tag{17}$$

Now, $\frac{P(w_i)}{P(w_{i-1})}$ is independent of the clustering used. Therefore, for selection of the best clusters, it is sufficient to try to maximize $\prod_{i=1}^{N} P(w_{i-1} \mid c_i)$. This is very convenient since it is exactly the opposite of what was done for conditional clustering. It means that we can use the same clustering tool for both, and simply switch the order used by the program used to get the raw counts for clustering. We give more details about the clustering algorithm in Appendix B.

5. Results and Discussion

In this section, we will report our main experiments. In Section 5.1, we will describe the text corpus we used. In Section 5.2, we will compare the performance of word-based trigram models with cluster-based *n*-gram models. We will give perplexity results of cluster-based *n*-gram models alone, as well as of combined models, where the cluster-based *n*-gram models were interpolated with word-based *n*-gram models. In Section 5.3, we will present a fairly thorough comparison of different techniques for using clusters in language model compression. We will then show that our novel clustering techniques can produce much smaller models at a given level of perplexity.

5.1 Corpora

We performed our experiments on both Chinese and Japanese text corpora. In both cases, we built language models on training data sets of medial size. We performed parameter optimization on a separate set of heldout data, and performed testing on a set of test sets. None of the three data sets overlapped. Out-of-vocabulary words were not included in perplexity computations.

For the Chinese corpus, we used the IME corpus for language model training. It is a balanced corpus, and it exhibits great variety in domain as well as in style. It was collected from the Microsoft input method editor (IME – a software layer that converts keystrokes into Chinese character) tasks. It consists of 11 million characters (or 7 million words after word segmentation). We used 10,000 words for heldout data, and 20,000 words for testing data. The heldout and test data set were every 50th sentence from two non-overlapping sets of an independent open test set. The open test set was carefully designed, and contains approximately half a million characters that have been proofread and balanced among domains, styles, and time [Gao *et al*., 2001]. The lexicon we used is defined by Chinese linguists, with 50,180 entries. The experiments on the Chinese corpus were fairly open tests since we used heterogeneous (in terms of domain and style) data sets from different sources for language model training and testing. Thus, we assumed that problems due to data sparseness and training-test mismatch would be relatively serious.

For experiments on Japanese language modeling, we used a subset of the Nikkei newspaper corpus. In particular, we used the most recent ten million words of the Nikkei corpus for training. As in the Chinese case, we used 10,000 words for heldout data, and 20,000 words for testing data. The heldout and test data sets were every 50th sentence from two non-overlapping sets, taken from another section of the Nikkei corpus. The lexicon we used contains 180,187 Japanese words. The experiments on the Japanese corpus were more like closed tests since we used homogeneous (at least in terms of style) data sets from the same corpus for language model training and testing. We then assumed that data sparseness and training-test mismatch would be less serious than they would be for the Chinese corpus. We also assumed that the Japanese lexicon was far more complete than the Chinese one. A certain number of the entries in the Japanese lexicon are expressions (e.g. of time and date).

Using the abovementioned Chinese and Japanese text corpora, we sought to test the robustness of our clustering techniques for different languages, corpora, and word sets (e.g. lexicons).

5.2 Clustering for language model improvement

The techniques for finding clusters described in Section 4 were applied to the training corpus to determine suitable word clusters. The word clusters obtained were used to define a cluster-based trigram model and to compute the perplexity on the test sets.

In the experiments, the clustering technique we used created a binary branching tree with words at the leaves. By cutting the tree at a certain level, it was possible to achieve a wide variety of different numbers of clusters. For instance, if the tree was cut after the 8th level, there would be roughly 2^8=256 clusters. Since the tree would not be balanced, the actual number of clusters could be somewhat smaller. Therefore, in what follows, we will use the level of the tree to represent approximately the number of clusters, such as 2^1, 2^2, 2^3, etc. Many more details about the clustering techniques used are given in Appendix B.

5.2.1 Using cluster-based trigram models alone

In the first series of experiments, we used the traditional cluster-based trigram model of Equation (6) to compute the perplexity. The results are shown in Table 1 for the Chinese and the Japanese corpora. For the sake of comparison, the perplexities of the word trigram models are included. In addition, the perplexities of several human defined word clusters sets are shown as well. These include (1) the 28 POS tags of the Chinese corpus [Zhou, 1996] and (2) the 1428 semantic clusters of the Chinese corpus, which were taken from "同义词词林" (TongYiCi CiLing), a widely used Chinese thesaurus [Mei, 1983]. As shown in Table 1, the perplexity was drastically decreased by increasing the number of word clusters. The best results on both Chinese and Japanese corpora are still the word-based trigram values. It turns out that human defined clusters work much worse than automatically derived clusters with similar numbers of word clusters. The results are consistent with those of Ney *et al.* [1994], who observed that for small amounts of training data (100,000 words), hand clustering outperformed automatic clustering, but that for larger amounts (1.1 million words), automatic clustering was better.

Notice that although the perplexity of the hand clustering model is much higher than the perplexity of the automatic clustering model, this does not mean that human defined clusters are unreasonable or worse than automatically derived clusters. The two cluster sets were generated by different criteria and motivations. Hand clustering is usually based on semantic/syntactic similarity, while automatic clustering uses the perplexity measurement directly. Therefore, the former is more widely used for knowledge systems, such as spoken language understanding, while the latter is good for statistical systems, such as speech recognition. As shown in table 4, although most of the automatically derived clusters look reasonable, there are also clusters which are difficult to interpret from a linguistic point of view.

Table 1. *Test set perplexities with cluster-based trigram models.*

Number of clusters	Chinese	Japanese
2ˆ5	644.31	346.05
2ˆ6	542.13	268.92
2ˆ7	464.76	223.70
2ˆ8	405.92	194.26
2ˆ9	358.57	172.63
2ˆ10	322.13	155.39
28 (POS clusters)	1038.56	––––––
1428 (semantic clusters)	676.87	––––––
Word trigram	242.74	106.33

5.2.2 Using combined models

In the second series of experiments, we used the combined models of Equation (9), where the cluster-based trigram model is linearly interpolated with the word-based trigram model. The interpolation constant λ is optimized on heldout data. The results are shown in Table 2. We still used word-based trigram models as baseline systems. It turns out that combined models consistently outperform baseline models. Unlike the case shown in the Table 1, the perplexity is decreased slowly at first by increasing the number of word clusters. We thus have an optimum at about 2^9 clusters for both the Chinese and the Japanese corpus. Beyond these numbers, the perplexity increases slightly again. Depending on the corpus, we have different levels of perplexity reduction: about 3% for the Chinese corpus (at 2^9 clusters), and more than 10% for the Japanese corpus (at 2^9 clusters).

Table 2. *Test set perplexities with combined trigram models.*

Number of clusters	Chinese	Japanese
2ˆ5	236.01	100.01
2ˆ6	235.02	98.21
2ˆ7	234.21	96.68
2ˆ8	233.53	95.73
2ˆ9	233.42	95.41
2ˆ10	234.11	95.66
2ˆ11	234.81	96.72
2ˆ12	235.53	97.60
2ˆ13	236.58	99.58
Word trigram	242.74	106.33

5.2.3 Using higher-order *n*-gram models

While trigram approximation has been proven, in practice, to be reasonable, there is disagreement about whether longer contexts can be helpful. This has led to research on using *n*-gram models in which $n>3$, called higher-order *n*-grams. Most of the previous experiments with higher-order *n*-grams showed little improvement because of the data sparseness problem. For example, [Goodman, 2001] showed that even using a very large corpus for *n*-gram model training (e.g. 280 million words), very small improvements occurred for *n*-gram models, where *n* is larger than 5. Clustering is an alternative way of dealing with the data sparseness problem besides smoothing. It was, thus, interesting to explore the effectiveness of cluster-based higher-order *n*-gram models.

We performed the third series of experiments on the relationship between cluster-based *n*-gram order and perplexity. We fixed the number of clusters at 2^8, and built a series of *n*-gram models, with *n* ranging from 2 to 20. The cluster-based higher-order *n*-gram models were then linearly interpolated with normal word-based trigram models. The perplexity results are shown in Table 3. We can see that although we used training corpora of medial size, improvement still occurred even for very high order *n*-gram models. After 10-gram models were used, depending on the corpus, we obtained approximately 10% perplexity reduction for the Chinese corpus, and obtained more than 11% perplexity reduction on the Japanese corpus. It then turns out that clustering works significantly better with higher-order *n*-gram models than the traditional smoothing methods as described in [Chen and Goodman, 1999].

Table 3. *Test set perplexities with cluster-based higher-order* n*-gram models.*

Order of cluster-based *n*-gram model	Chinese	Japanese
2	238. 93	99. 00
3	233. 53	95. 73
4	230. 17	93. 74
5	226. 79	93. 62
6	224. 52	93. 91
7	223. 01	94. 19
8	221. 37	94. 33
9	220. 05	94. 47
10	219. 44	94. 47
12	219. 14	94. 53
20	219. 13	94. 53
word trigram	242. 74	106. 33

5.2.4 Analysis of words in clusters

We divided the 50,180-entry Chinese lexicon into 2^8 clusters by means of automatic clustering. The number of words in each cluster varied greatly from 0 to more than 2000. Table 4 gives 11 examples of word clusters. For each cluster from **A** to **C**, we randomly selected 10 two-character Chinese words, and removed those words that occurred less than 10 times in the training corpus. For each remaining cluster shown in table 4, we give the top 15 to 30 two-character Chinese words with the highest frequency (at lest 10 times) in the training corpus.

We can see that most of the words in each cluster belong to the same syntactic class, namely, verbs for cluster **A**, nouns for clusters **B** and **C**, etc. Furthermore, there are some semantic similarities between the words in a cluster. The majority of the words in cluster **A** are verbs expressing some kind of motion, some of the words in cluster **B** are titles, and some of the words in cluster **C** are games. There are also words which appear to be in the wrong cluster: words like “earth” and “banquet” are not games, and words like “parsimony” and “mournful” are not verbs. Although most of the clusters look reasonable, there are also clusters that are difficult to interpret from a linguistic point of view. The other 8 clusters, which contain only high frequency words, look quite reasonable. It turns out that, given a sufficient large training corpus, the degree to which the clusters capture both syntactic and semantic aspects of Chinese is quite impressive although they were constructed from nothing more than counts of bigrams.

Table 4. *Most frequent words of some sample clusters from the Chinese corpus.*

Cluster	Words
A	走(walk), 飞跑(run), 奔腾(rush), 高攀(climb up), 推翻(overset), 跳动(jump), 流淌(flow), 吝惜(parsimony), 凄然(mournful), …
B	老师(teacher), 先生(sir), 小姐(miss), 同志(comrade), 父亲(father), 母亲(mother), 讨伐(crusade against), 发誓(promise), …
C	篮球(basketball), 棒球(baseball), 乒乓球(ping-pang), 铅球(shot), 地球(earth), 宴会(banquet), …
D	进行(conduct), 建立(build), 提出(bring forth), 实现(accomplish), 取得(gain), 提供(provide), 出现(advent), 得到(annex), 形成(form), 发生(occur), 发挥(develop), 产生(accrue), 完成(complete), 获得(get), 发表(publish), 创造(create), 召开(convene), 出席(attend), 所有(all), …
E	继续(keep on), 再次(once more), 重新(over again), 坚决(determined), 第一次(the first time), 多次(several times), 经常(often), 纷纷(one after another), 突然(suddenly), 立即(at once), 刚刚(a moment ago), 逐渐(gradually), 尽快(as soon as possible), 主动(active), 从中(from it), 亲自(personally), 彻底(thoroughly), 提前(advanced), 反复(again and again), 马上(immediately), …
F	汽车(automobile), 石油(petroleum), 建筑(architecture), 制造(manufacture), 加工(process), 食品(food), 化学(chemistry), 化工(chemical engineering), 机械(mechanics), 本地(native), 广告(advertisement), 航空(aerial), 制作(manufacture), 航天(spaceflight), 示范(demonstration), 电力(power), 服装(garment), 纺织(spinning), 钢铁(steel and iron), 走私(smuggle), …

G	重要(important), 主要(dominant), 群众(mass), 一定(certain), 基本(elementary), 重大(fatal), 实际(practical), 一切(the whole), 高度(high), 人类(mankind), 一般(general), 具体(concrete), 根本(basic), 自然(natural), 核心(kernel), 特殊(special), 自身(oneself), 客观(objective), 各自(respective), 唯一(unique), 最好(best), 自我(self), 周围(surrounding), 军人(soldier), 绝对(absolute), 历史性(historic), 彼此(one another), 最低(lowest), …
H	广州(Guangzhou), 贫困(poor), 深圳(Shenzhen), 天津(Tientsin), 纽约(New York), 南京(Nanking), 厦门(Xiamen), 重庆(Chongqing), 巴黎(Paris), 东北(northeast), 西安(Xi’an), 福州(Fuzhou), 长江(Yangtze River), 华盛顿(Washington), 东京(Tokyo), 成都(Chengdu), 大连(Dalian), 珠海(Zhuhai), 武汉(Wuhan), 沿海(coastal), 西南(southwest), 南方(south), 黄河(Yellow River), 整顿(put to order), 山区(mountain region), …
I	所谓(so-called), 称为(call), 誉为(call), 可谓(call), 叫做(call), 称之为(call), 致函(address a letter), 评为(call), 人称(namely), 发给(hand out), 称作(call), 素有(have), 号称(reputed), 鼓吹(promote by publicity), 当作(regard as), …
J	过去(past), 以后(after), 之后(later on), 当时(at that time), 一天(one day), 后来(later on), 如今(now), 为此(for the purpose), 另外(for the purpose), 当年(that year), 晚上(night), 不久(soon), 面前(in front), 之前(before), 身上(body), 这时(this time), 拒绝(reject), 中间(intermediate), 随后(later on), 那天(that day), …
K	积极(positively), 不断(constantly), 充分(adequately), 认真(seriously), 广泛(widely), 深入(deeply), 正确(correctly), 有效(availably), 真正(really), 逐步(stepwise), 健康(healthily), 明显(obviously), 迅速(promptly), 严格(sternly), 明确(explicitly), 顺利(smoothly), 普遍(generally), 热烈(warmly), 热情(passionately), 合理(reasonably), 及时(timely), 切实(practically), 更好(get better), 有力(strongly), 大大(greatly), 显著(significantly), 自觉(voluntarily), 相应(correspondingly), …

5.3 Clustering for language model compression

As shown in the last subsection, Equation (9) leads to better results (lower perplexities) than a simple trigram model does. But at the same time, the combined model is larger, since it includes both a cluster-based trigram model and a normal trigram model. In this subsection, we will explain how we took memory constraints into consideration, and concentrated our experiments on using clustering for language model compression. We performed experiments on the three basic cluster-based trigram models described in Section 3, and we found that our novel clustering techniques could be combined with language model pruning methods to produce much smaller models at a given level of perplexity than could be produced using pruning methods without clustering.

Since we are seeking the correct balance between memory storage and perplexity, all experimental results below are presented in the form of size/perplexity curves. The size was measured as the total number of parameters of the language model: one parameter for each bigram and trigram one parameter for each normalization parameter α that was needed, and one parameter for each unigram. In the pruning experiments, bigrams and trigrams were both pruned, unigrams never were. This resulted in the smallest possible number of parameters being equal to the vocabulary size, e.g. 50,187 unigrams for Chinese models, and 180,187 unigrams for Japanese models.

In our experiments described below, we used Stolcke's [1998] pruning method to produce a series of language models of different sizes. This method is an entropy-based cutoff method, and can be considered an extension of the work of Seymore and Rosenfeld [1996] and of Kneser [1996]. The basic idea is to remove as many "useless" probabilities as possible, and at the same time to keep the perplexity increase as small as possible. This is achieved by examining the weighted relative entropy or Kullback-Leibler distance between each probability $P(w|h)$ and its value from the backoff distribution, $\overline{P}(w|\overline{h})$:

$$D(P(w\mid h)\,\|\,\overline{P}(w\mid \overline{h})) = P(w\mid h) * \log \frac{P(w\mid h)}{\overline{P}(w\mid \overline{h})}, \tag{18}$$

where $\overline{h}$ is the reduced history. When the Kullback-Leibler distance is small, the backoff probability is a good approximation, and the probability $P(w|h)$ does not carry much additional information and can be deleted. The Kullback-Leibler distance was calculated for each n-gram entry, and we removed entries and reassigned the deleted probability mass to backoff mass for any n-gram entry whose deletions increased the Kullback-Leibler distance by less than a specified threshold value. Compared to the traditional count cutoff methods, Stolcke pruning performed a little better [Goodman and Gao, 2000]. More importantly, the Stolcke method could prune a model to a specific size, simply by finding the threshold level that resulted in a model of that size. For all the models, we used a smoothing method called *modified absolute discounting* for backoff. We give more details about Stolcke pruning and modified absolute discounting in Appendix A.

We then performed a number of experiments to compare our different models. In these experiments, the baseline system was the word-based trigram model.

5.3.1 Predictive clustering

We first used predictive clustering of Equation (10). The results are shown in Figures 1 and 2. It turns out that we got the best result at about 2^6 clusters for both the Chinese and Japanese corpora. Depending on the corpus, compared to the baseline systems, at the same size, we got a maximum 6.6% perplexity reduction for the Chinese corpus, and a maximum 5.1% perplexity reduction for the Japanese corpus; at the same perplexity, we got a maximum 54% size reduction for the Chinese corpus, and a maximum 57% size reduction for the Japanese corpus. We notice that for these two corpora, although we got the best result at 2^6 clusters for both of them, the results at other numbers of clusters (e.g. 2^4, 2^7) were very different. For the Chinese corpus, all the predictive clustering models performed about the same. For the Japanese corpus, models at larger numbers of clusters performed much better than models at small numbers of clusters (e.g. 2^4). In general, with our clustering, when there was only a small amount of training data, the clusters became less useful. Perhaps this was because there was a more serious data sparseness problem for the Chinese corpus, and many parameters were out of training, thus larger clusters did not bring benefits. As for the Japanese corpus, the data sparseness problem was much less serious, so a large number of clusters led to significant perplexity reduction.

We also tried to set different pruning threshold values for the two components of the predictive clustering models. We could not obtain any improvement. Therefore, in what follows, we will always assume that we used the same pruning threshold value for both components in the predictive clustering and combined clustering models.

5.3.2 Conditional clustering

We used the conditional clustering of Equation (11). As shown in Figures 3 and 4, the results for the two languages are qualitatively very similar. The performance was consistently improved by increasing the number of clusters. But no conditional clustering model was superior to the baseline model. This is not surprising because the conditional clustering model always discards information for predicting words, and even with smoothing, it does not bring any additional benefits.

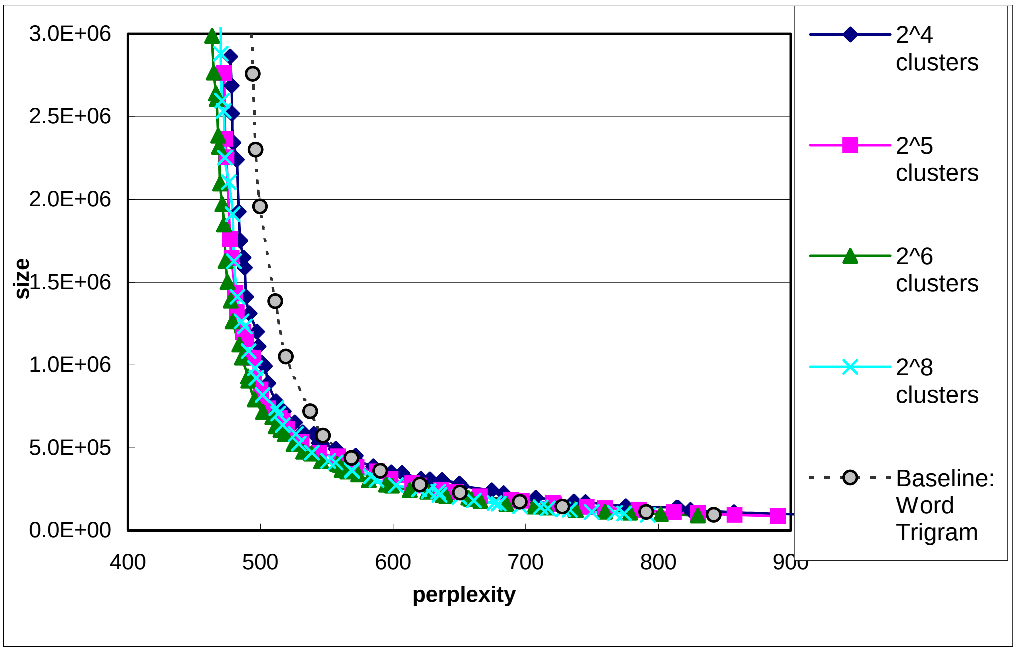

Figure 1 *Comparison of predictive models applied with different numbers of clusters to the Chinese corpus.*

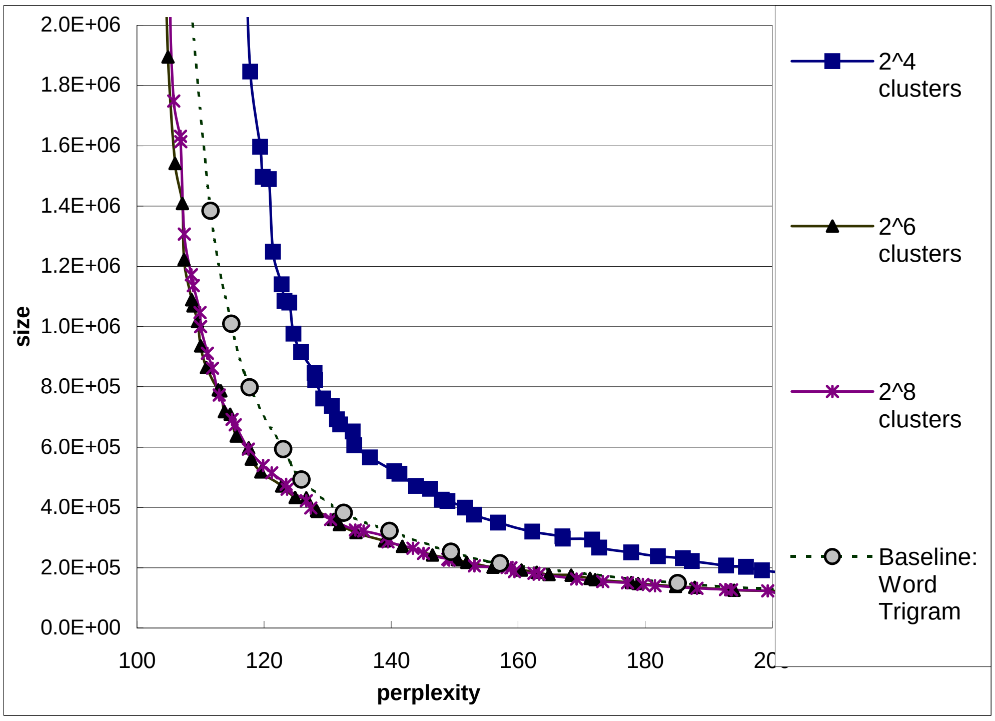

Figure 2. *Comparison of predictive models applied with different numbers of clusters to the Japanese corpus.*

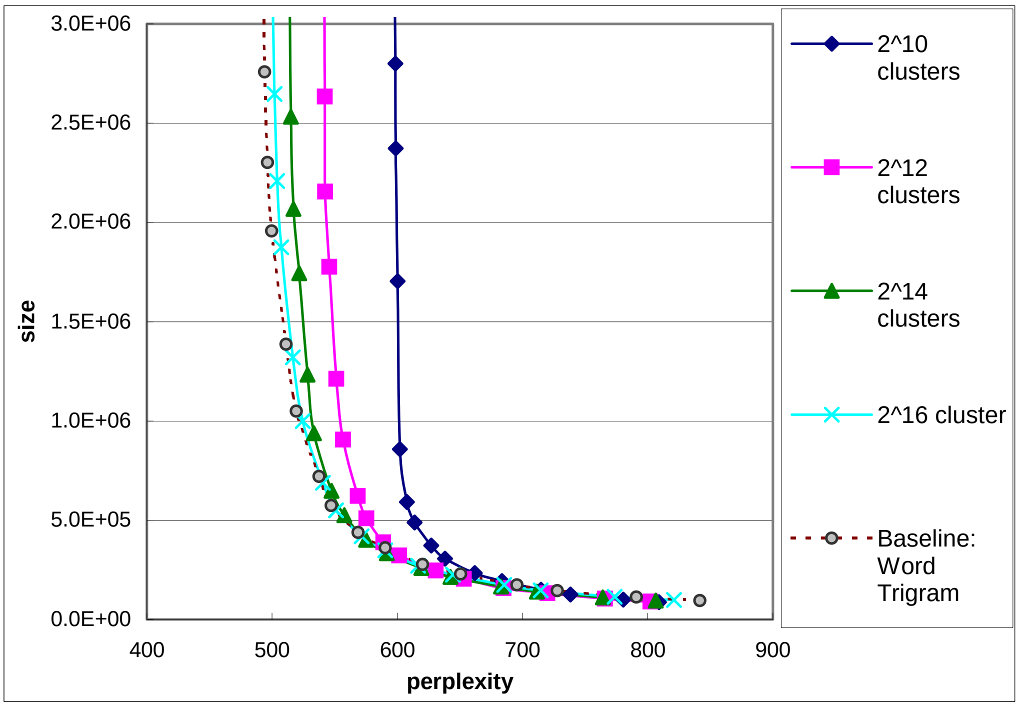

Figure 3 *Conditional models applied with different numbers of clusters to the Chinese corpus.*

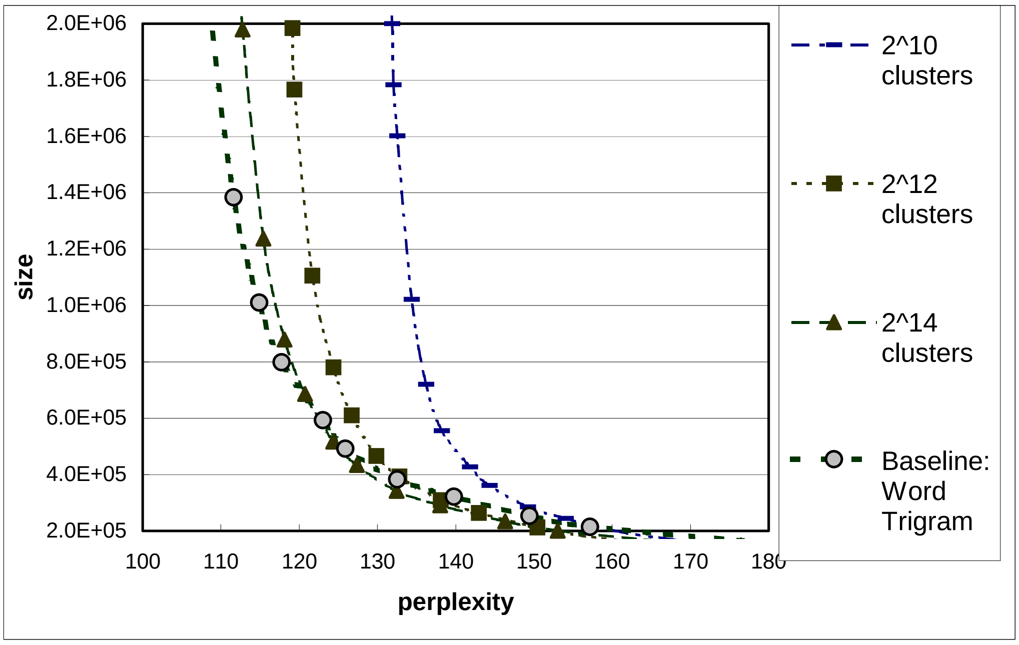

Figure 4 *Comparison of conditional models applied with different numbers of clusters to the Japanese corpus.*

5.3.3 Combined clustering

We also used the combined clustering of Equation (12). As mentioned earlier, we can use different numbers of cluster for predictive clusters and conditional clusters. This leads to a very large number of possible parameter settings. We presented detailed analysis of the parameter settings of the combined clustering model in [Goodman and Gao, 2000]. In this paper, we will only report the results of some sample parameter settings.

For the Chinese corpus, as shown in Figure 5, we set the number of predictive clusters to 2^4, 2^6, and 2^8, and set the number of conditional clusters to 2^12, 2^14, and 2^16. We then built a large number of models. Rather than graph all the points of all the models, we show only the outer envelope of the points for each number of predictive clusters in Figure 5. That is, if for a model with a given number of predictive clusters, there was some other point with the same number of predictive clusters (and perhaps a different number of conditional clusters) with both lower perplexity and smaller size than the first model, then we did not graph the first, worse point. We show the outer envelope of the size/perplexity curves of 2^4, 2^6, and 2^8 predictive clusters.

For the Japanese corpus, as shown in Figure 6, we do not show the outer envelopes as in Figure 5. Instead, we show results of the top three best parameter settings we obtained; for instance, (2^4, 2^12) represents the combined cluster-based trigram model with 2^4 predictive clusters and 2^12 conditional clusters.

It turns out that, for the Japanese corpus, the best combined clustering models outperformed the baseline model. At small model sizes, we got the best result at 2^14 conditional clusters and 2^6 predictive clusters. At large model sizes, we got the best result at 2^12 conditional clusters and 2^4 predictive clusters. We achieved the maximum 6.5% perplexity reduction at the same size, and the maximum 40% size reduction at the same perplexity. But for the Chinese corpus, no improvement over the baseline model was achieved until we used models with very large numbers of conditional clusters. This is not difficult to explain. Recall that predictive clustering is a special case of combined clustering. Actually, in most combined clustering models for Chinese, it turns out to be no less optimal to use conditional clusters than the vocabulary size, i.e., no conditional clustering.

Now, consider the IBM clustering of Equation (6), which is a special case of the combined clustering model. As shown in Figure 6, the performance is by far the worst, roughly an order of magnitude worse than the performance of the other approaches. We hypothesized that this was because the IBM model throws away too much useful information. We thus tried a variation on the IBM model:

$$\lambda P(w_i \mid w_{i-2}w_{i-1}) + (1-\lambda)P(w_i \mid c_{i-2}c_{i-1}c_i) \times P(c_i \mid c_{i-2}c_{i-1}) \,. \qquad (20)$$

This model is just like the standard IBM model, but it also conditions the probability of the word on the previous clusters. We compared this model with a standard IBM model. The results are shown in Tables 5 and 6. It turns out that, for the Chinese corpus, models in the form of Equation (20) consistently outperformed the standard IBM models (e.g. we achieved 4% perplexity reduction at 2^9 clusters), while for the Japanese corpus, they worked about the same. Notice that in these experiments, no pruning was done.

We summarize the results of all the experiments described in this subsection in Figures 7 and 8. It is clearly seen that our novel clustering techniques produce much smaller models than do previous methods (i.e. baseline systems) at the same perplexity level. In addition, several more conclusions can be drawn:

1. Conditional clustering did not help for either the Chinese or the Japanese corpus since it always discards information.
2. For closed tests on homogeneous text corpora (e.g. the Japanese corpus), both combined clustering and predictive clustering outperformed the baseline system consistently. Combined clustering is better at small model sizes, while predictive clustering is better at larger sizes.
3. For open tests on heterogeneous text corpora (e.g. the Chinese corpus), predictive clustering outperformed the baseline system consistently. Although the results presented in this paper show that combined clustering achieved no improvement, in [Goodman and Gao, 2000], we showed that with more sophisticated techniques, it appears possible to make combined clustering better than predictive clustering.

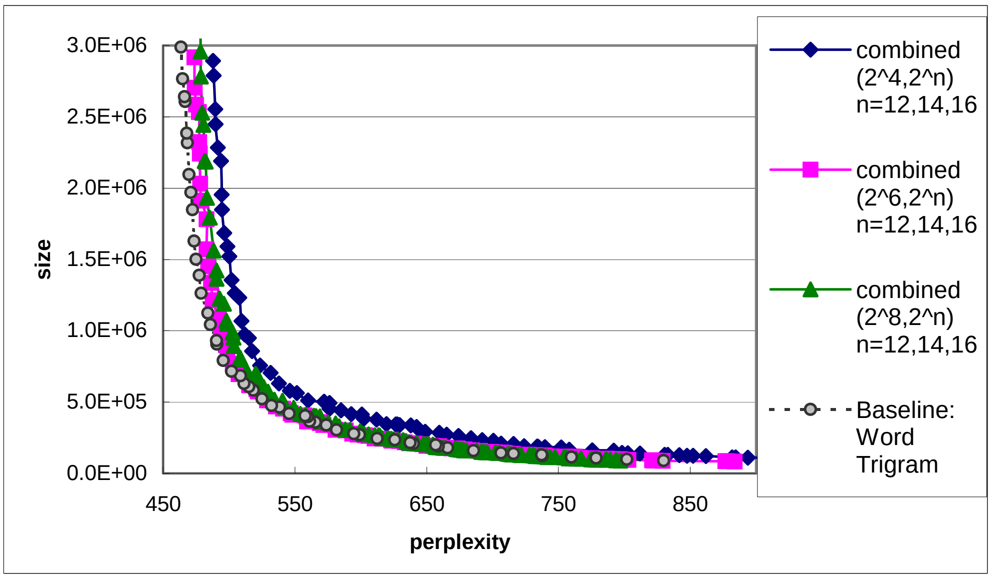

Figure 5 *Comparison of combined clustering models applied with different numbers of clusters to the Chinese corpus.*

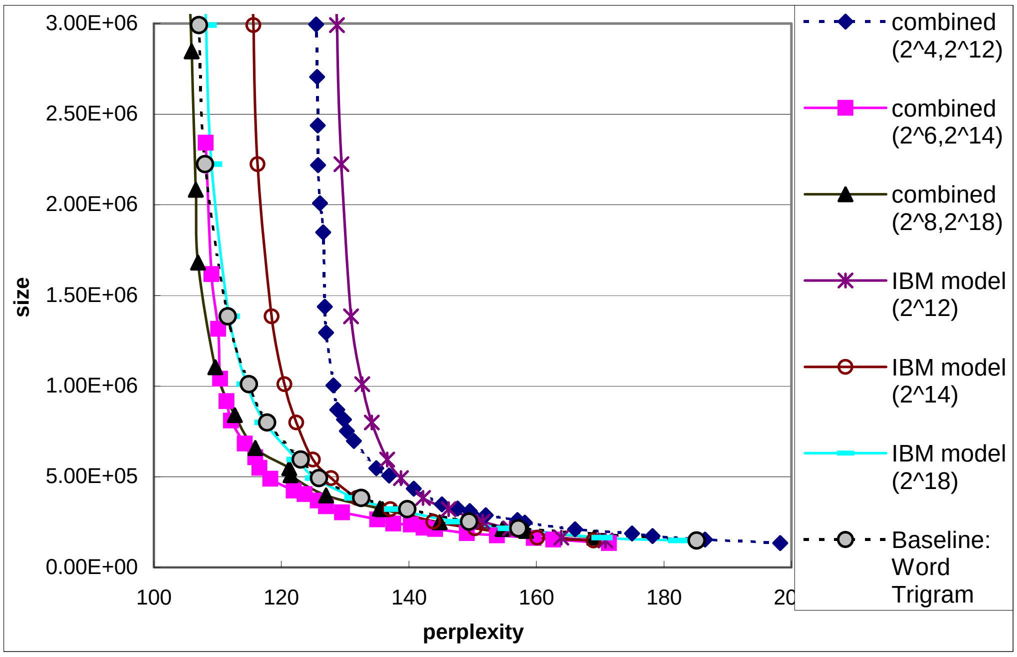

Figure 6 *Comparison of combined clustering models applied with different numbers of clusters to the Japanese corpus and the IBM model.*

Table 5. *Comparison of different combined trigram models applied to the Chinese corpus.*

Number of clusters	Equation (9)	Equation (20)
2ˆ6	235. 02	226.65
2ˆ7	234. 21	224.65
2ˆ8	233. 53	224.29
2ˆ9	233. 42	224.99
2ˆ10	234. 11	226.53
2ˆ11	234. 81	228.26
2ˆ12	235. 53	230.95
2ˆ13	236. 58	234.78
word trigram	242. 74	242.74

Table 6. *Comparison of different combined trigram models applied to the Japanese corpus.*

Number of clusters	Equation (9)	Equation (20)
2ˆ6	98. 21	97.06
2ˆ7	96. 68	96.42
2ˆ8	95. 73	96.33
2ˆ9	95. 41	96.41
2ˆ10	95. 66	96.82
2ˆ11	96. 72	97. 57
2ˆ12	97. 60	98. 50
2ˆ13	99. 58	100. 52
word trigram	106. 33	106. 33

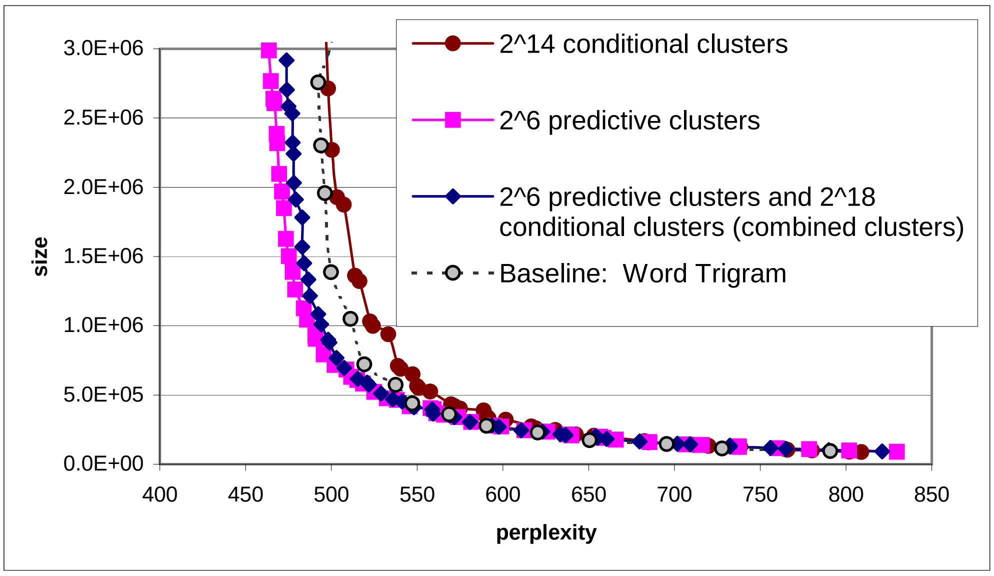

Figure 7 *Summary of the results obtained by applying clustering models to the Chinese corpus.*

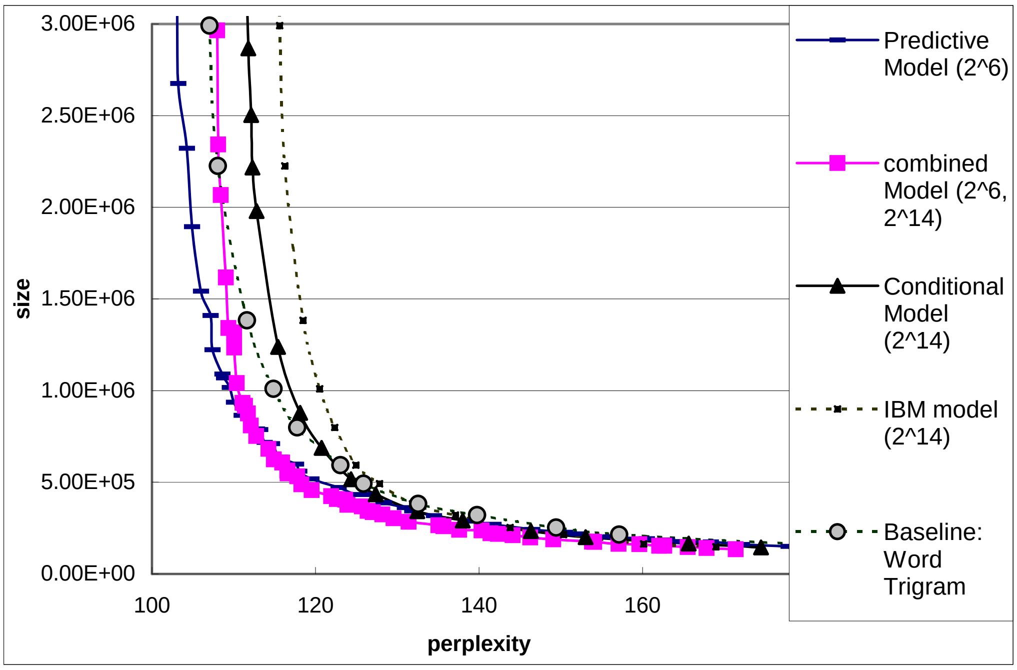

Figure 8 *Summary of the results obtained by applying clustering models to the Japanese corpus.*

6. Conclusion

Cluster-based *n*-gram models are variations on traditional word-based *n*-gram models. They attempt to make use of the similarities between words. In this paper, we have presented an empirical study on clustering techniques for Asian language modeling. While the majority of the previous research on word clustering has focused on how to get the best clusters, we have concentrated our research on the best way to use the clusters. We have studied in detail three cluster-based *n*-gram models, namely, *predictive clustering*, *conditional clustering*, and *combined clustering*. In our experiments, clustering was used to improve the performance (i.e. perplexity) of language models as well as to compress language models. We performed experimental tests on a Japanese newspaper corpus of more than 10 million words, and on a Chinese mixed-domain corpus of more than 7 million words. Results show that our novel techniques worked much better than previous methods. They not only showed better performance when interpolated with normal *n*-gram models, but could also be combined with Stolcke pruning to produce models much smaller than unclustered models with the same perplexity level.

Most language modeling improvements reported previously required significantly more space than the normal trigram baseline model, or had higher perplexity. Their practical value is questionable. In this paper, we have proposed a technique that results in lower perplexity than the traditional trigram models do at every memory size. In other research [Gao *et al.*, 2001], we have shown that cluster-based models of this form can be applied effectively to pinyin to Chinese character conversion. One area we consider promising for future research is the combination of human defined and automatically derived clustering. While human defined clusters alone generally work worse than automatically derived clusters, there has been little research on their combination. It is an open question whether such a combination can lead to further improvements.

Acknowledgements

We would like to thank Prof. Changning Huang, Dr. Ming Zhou, and other colleagues at Microsoft Research, Yoshiharu Sato, and Hiroaki Kanokogi at the Microsoft (Japan) IME group, for their help in developing the ideas and implementation presented in this paper. We would also like to thank Jiang Zhu and Miyuki Seki for their help in our experiments and for providing Chinese and Japanese text corpora.

References

Bai, S., Li, H., Lin, Z., and Yuan, B. "Building class-based language models with contextual statistics." In *ICASSP-98*, 1998. pp. 173-176.

Bellegarda, J. R., Butzberger, J. W., Chow, Y. L., Coccaro, N. B., and Naik, D. "A novel word clustering algorithm based on latent semantic analysis." In *ICASSP-96*, 1996. pp. I172-I175.

Brown, P. F., Cocke, J., DellaPietra, S. A., DellaPietra, V. J., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. "A statistical approach to machine translation." *Computational Linguistics,* 16(2), 1990. pp. 79-85.

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

Chen, S. F., and Goodman, J. "An empirical study of smoothing techniques for language modeling." *Computer Speech and Language,* 13:359-394, October. 1999.

Church, K. "A stochastic parts program and noun phrase parser for unrestricted text." In *Proceedings of the Second Conference on Applied Natural Language Processing*, 1988. pp. 136-143.

Cutting, D. R., Karger, D. R., Pedersen, J. R., and Tukey, J. W. "Scatter/gather: A cluster-based approach to browsing large document collections." In *SIGIR 92*. 1992.

Gao, J., Goodman, J., Li, M., and Lee, K. F. "Toward a unified approach to statistical language modeling for Chinese." To appear in *ACM Transactions on Asian Language*

Information Processing. 2001. Draft available from http://www.research.microsoft.com/~jfgao

Goodman, J. “A bit of progress in language modeling.” Submitted to *Computer Speech and Language*. 2001. Draft available from http://www.research.microsoft.com/~joshuago

Goodman, J., and Gao, J. “Language model compression by predictive clustering.” *ICSLP-2000*, Beijing, October. 2000.

Heeman, P. “POS tags and decision trees for language modeling.” In *ACL-99*, 1999. pp. 129-137.

Heeman, P., and Allen, J. “Incorporating POS tagging into language modeling.” In *Eurospeech-97*, Ghodes, Greece, 1997. pp. 2767-2770.

Huang, X. D., Acero, A., and Hon, H. Spoken language processing. Prentice Hall PTR. 2001.

Issar, S., and Ward, W. “Flexible parsing: CMU’s approach to spoken language understanding.” In *Proceedings of the ARPA Spoken Language Technology Workshop*, pp. 53-58. 1994.

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

Jurafsky, D., and Martin, J. H. Speech and language processing. Prentice Hall.

Katz, S. M. 1987. “Estimation of probabilities from sparse data for the language model component of a speech recognizer.” *IEEE Transactions on Acoustics, Speech and Signal Processing*, ASSP-35(3):400-401, March. 2000.

Kernighan, M. D., Church, K. W., and Gale, W. A. “A spelling correction program based on a noisy channel model.” In *Proceedings of the Thirteenth International Conference on Computational Linguistics*, 1990. pp. 205-210.

Kneser, R. and Ney, H. “Improved clustering techniques for class-based statistical language modeling.” In *Eurospeech,* Vol. 2*,* 1993. pp. 973-976, Berlin, Germany.

Kneser, R. “Statistical language modeling using a variable context length.” Proc. ICSLP, volume 1, pages 494-497, Oct. 1996.

Maltese, B., and Mancini, F. “An automatic technique to include grammatical and morphological information in a trigram-based statistical language model.” In *ICASSP-92*, 1992. pp. I157-I160.

Manning, C. D., and Schutze, H. “Foundations of Statistical Natural Language Processing.” MIT Press, 1999. Cambridge, MA.

Mei, J. Z. Tongyici Cilin. Shanghai Cishu Publishing House, 1983. Shanghai.

Miller, D., Leek, T., and Schwartz, R. M. “A hidden Markov model information retrieval system.” In *Proc. 22nd International Conference on Research and Development in Information Retrieval*, Berkeley, CA, 1999, pp. 214-221.

Miller, J. W., and Alleva, F. “Evaluation of a language model using a clustered model back off.” In *ICASSP-97*, 1997. pp. 390-393.

Ney, H., Essen, U., and Kneser, R. "On structuring probabilistic dependences in stochastic language modeling." *Computer Speech and Language*, 8:1-38. 1994.

Niesler, T. R., Whittaker, E. W. D., and Woodland, P. C. "Comparison of part-of-speech and automatically derived category-based language models for speech recognition." In *ICASSP-98*, 1998. pp. I177-I180.

Pereira, F., Tishby, N., and Lee L. "Distributional clustering of English words." In *Proceedings of the 31st Annual Meeting of the ACL.* 1993.

Placeway, P., Schwartz, R., Fung, P., and Nguyen, L. "The estimation of powerful language models from small and large corpora." In *ICASSP-93*, 1993. II33-36.

Seymore, K. and Rosenfeld, R. "Scalable backoff language models", *Proc. ICSLP*, Vol. 1., 1996. pp.232-235, Philadelphia,

Srinivas, B. "Almost parsing techniques for language modeling." In *ICSLP-96*, 1996. pp. 1169-1172.

Stolcke, A. "Entropy-based Pruning of Backoff Language Models." In *Proc. DARPA News Transcription and Understanding Workshop*, Lansdowne, VA. 1998. pp. 270-274. See corrections at http://www.speech.sri.com/people/stolcke

Ueberla, J. P. "An extended clustering algorithm for statistical language models." *IEEE Transactions on Speech and Audio Processing*, 4(4): 313-316. 1996.

Ward, W., and Young, S. "Flexible use of semantic constraints in speech recognition." In ICASSP-93, 1993. pp. II49-50.

Yamamoto, H., and Sagisaka, Y. "Multi-class Composite N-gram based on Connection Direction." In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, May, 1999. Phoenix, Arizona.

Yang, Y. J., *et al*., "An intelligent and efficient word-class-based Chinese language model for Mandarin speech recognition with very large vocabulary." In *ICSLP-94*, Yokohama, Japan, 1994. pp. 1371-1374.

Zhou, Q. "Phrase bracketing and annotating on Chinese language corpus." Ph.D. dissertation. Beijing University. 1996.

Zue, V. W. Navigating the information superhighway using spoken language interfaces. *IEEE Expert*, vol. 10, no. 5, pp. 39-43, October, 1995

A. Methods of Trigram Training

We will describe methods for language model training below. These include the modified absolute discounting smoothing method and Stolcke’s entropy-based pruning method.

Absolute Discounting

Trigram Language models make the assumption that the probability of a word depends only on the identity of the immediately two preceding words, say $P(w_i|w_1\ w_2...\ w_{i-1}) \approx P(w_i|w_{i-2}\ w_{i-1})$.

Smoothing is used to address the problem of data sparseness. Experimental results show that a novel variation of absolute discounting, Kneser-Ney smoothing, consistently outperforms all others [Chen and Goodman, 1999]. However, because Kneser-Ney smoothing is less commonly used, slightly more difficult to implement, and, we suspect, may not work as well when pruning is done, we used a slightly different technique in this research, modified absolute discounting. First, we will describe basic absolute discounting. Letting D represent a discount, we set the probability as follows:

$$P_{absolute}(w_i \mid w_{i-2}w_{i-1}) = \begin{cases} \dfrac{\max(C(w_{i-2}w_{i-1}w_i) - D, 0)}{C(w_{i-2}w_{i-1})} & if\ C(w_{i-2}w_{i-1}w_i) > 0 \\ \alpha(w_{i-2}w_{i-1}) P_{absolute}(w_i \mid w_{i-1}) & otherwise \end{cases} . \quad (21)$$

$\alpha(w_{i-2}w_{i-1})$ is defined in such a way that the probabilities sum to 1:

$$\alpha(w_{i-2}w_{i-1}) = \frac{1 - \sum_{w_i : C(w_{i-2}w_{i-1}w_i) > 0} P_{absolute}(w_i \mid w_{i-2}w_{i-1})}{1 - \sum_{w_i : C(w_{i-2}w_{i-1}w_i) > 0} P_{absolute}(w_i \mid w_{i-1})} . \quad (22)$$

The trigram backs off to the bigram, and the bigram backs off to the unigram. The unigram does not need to be smoothed although it can be smoothed with the uniform distribution. In practice, different D’s are used for the bigram and trigram.

A further improvement is to use multiple discount D. Taking the trigram as an example, D_1 stands for counts $C(w_{i-2}w_{i-1}w_i) = 1$, D_2 for $C(w_{i-2}w_{i-1}w_i) = 2$, and a final one, D_3 for $C(w_{i-2}w_{i-1}w_i) \geq 3$. Chen and Goodman [1999] introduced an estimate for the optimal D for absolute discounting smoothing as a function of training data counts[1]. In practice, we can use Equation (23) to Equation (26) to approximately estimate the optimal values for D_1, D_2, and

[1] Thanks to Ries, K.

D_3:

$$Y = \frac{n_1}{n_1 + 2n_2}, \tag{23}$$

$$D_1 = 1 - 2Y\frac{n_2}{n_1}, \tag{24}$$

$$D_2 = 2 - 3Y\frac{n_3}{n_2}, \tag{25}$$

$$D_3 = 3 - 4Y\frac{n_4}{n_3}. \tag{26}$$

where n_1, n_2, n_3, and n_4 are total numbers of trigrams with exactly one, two, three, and four counts, respectively.

Notice that for the experiments conducted in this study, we did not use this approximation, but instead optimized the discounts on heldout data. This led to very limited improvement.

Entropy-based Pruning

Stolcke [1998] proposed a criterion for pruning *n*-gram language models based on the relative entropy between the original and the pruned model. The relative entropy measure can be expressed as a relative change in training data perplexity. All *n*-grams that change perplexity by less than a threshold are removed from the model.

Formally, let P denote the trigram probabilities assigned by the original model, say $P=P(w_i|w_{i-2}w_{i-1})$, and let $P'= P(w_i|\ w_{i-1})$ denote the probabilities in the pruned model, assuming that we have pruned the trigram probability. Then, the relative entropy between the two models is

$$\begin{aligned} D(P \,\|\, P') = -P(w_{i-2}w_{i-1})\{&P(w_i \mid w_{i-2}w_{i-1})[\log P(w_i \mid w_{i-1}) + \log\alpha'(w_{i-2}w_{i-1}) - \log P(w_i \mid w_{i-2}w_{i-1})] \\ &+[\log\alpha'(w_{i-2}w_{i-1}) - \log\alpha(w_{i-2}w_{i-1})] \sum_{w_i:C(w_i,w_{i-2}w_{i-1})=0} P(w_i \mid w_{i-2}w_{i-1})\} \end{aligned} \tag{27}$$

where $\alpha'(w_{i-2}w_{i-1})$ is the revised backoff weight after pruning. Recall that $\alpha(w_{i-2}w_{i-1})$ is estimated by Equation (22), and that $\alpha'(w_{i-2}w_{i-1})$ is obtained by dropping the term for the pruned trigram ($w_{i-2}w_{i-1}w_i$) from the summation in both numerator and denominator.

B. Clustering Algorithm

There is no shortage of techniques for generating clusters, and there appears to be little evidence that different techniques that optimize the same criterion result in significantly different quality of clusters. We note, however, that different algorithms may require significantly different amounts of run time. We used several techniques to speed up our clustering significantly.

The basic criterion we followed was to minimize entropy. In particular, we assumed that the model we were using was of the form $P(z|Y)$; we wanted to find the placement of words y into clusters Y that minimized the entropy of this model. This is typically done by swapping words between clusters whenever such a swap reduces the entropy.

The first important approach we took to speeding up clustering was a top-down approach. We note that agglomerative clustering algorithms – those which merge words bottom up – may require significantly more run time than top-down, splitting algorithms. Thus, our basic algorithm is top-down. However, in the end, we sometimes perform four iterations of swapping all words between all clusters. Notice that for the experiments reported in this paper, we used the basic top-down algorithm without swapping.

Another technique we used is Buckshot [Cutting *et al.*, 1992]. The basic idea is that even with a small number of words, we are likely to have a good estimate of the parameters of a cluster. Therefore, we proceeded top down, splitting clusters. When we were ready to split a cluster, we randomly picked a few words, and put them into two random clusters, and then swapped them in such a way that entropy was decreased, until convergence occurred (no more decrease could be achieved). Then we added a few more words, typically $\sqrt{2}$ more, put each one into the best bucket, and then swapped again until convergence occurred. This was repeated until all words in the current cluster had been added and split. We haven't tested this particularly thoroughly, but our intuition is that it should lead to large speedups.

We used one more important technique that speeds up computations, adapted from an earlier work by [Brown *et al.*, 1992]. We attempted to minimize the entropy of our clusters. Let v represent words in the vocabulary, and let W represent a potential cluster. We minimize

$$\sum_v C(Wv)\log P(v\,|\,W)\,.$$

The inner loop of this minimization considers adding (or removing) a word x to cluster W. What will the new entropy be? On its face, this would appear to require computation proportional to the vocabulary size to re-compute the sum. However, letting the new cluster $W + x$ be called X,

$$\sum_v C(Xv)\log P(v\,|\,X) = \sum_{v|C(xv)\neq 0} C(Xv)\log P(v\,|\,X) + \sum_{v|C(xv)=0} C(Xv)\log P(v\,|\,X)\,. \tag{28}$$

The first summation in Equation 28 can be computed relatively efficiently, in an amount of time proportional to the number of different words that follow x; it is the second summation that needs to be transformed:

$$\begin{aligned}\sum_{v|C(xv)=0} C(Xv)\log P(v\mid X) &= \sum_{v|C(xv)=0} C(Wv)\log\left(P(v\mid W)\frac{C(W)}{C(X)}\right)\\ &= \sum_{v|C(xv)=0} C(Wv)\log P(v\mid W) + \left(\log\frac{C(W)}{C(X)}\right)\sum_{v|C(xv)=0} C(Wv)\,.\end{aligned} \tag{29}$$

Now, notice that

$$\sum_{v|C(xv)=0} C(Wv)\log P(v\mid W) = \sum_{v} C(Wv)\log P(v\mid W) - \sum_{v|C(xv)\neq 0} C(Wv)\log P(v\mid W)\,, \tag{30}$$

and that

$$\sum_{v|C(xv)=0} C(Wv) = \left(C(W) - \sum_{v|C(xv)\neq 0} C(Wv)\right), \tag{31}$$

Substituting Equation 30 and 31 into Equation (29), we get

$$\begin{aligned}&\sum_{v|C(xv)=0} C(Xv)\log P(v\mid X)\\ &= \sum_{v} C(Wv)\log P(v\mid W) - \sum_{v|C(xv)\neq 0} C(Wv)\log P(v\mid W) + \left(\log\frac{C(W)}{C(X)}\right)\left(C(W) - \sum_{v|C(xv)\neq 0} C(Wv)\right)\cdot\end{aligned} \tag{32}$$

Now, notice that $\sum_{v} C(Wv)\log P(v\mid W)$ is just the old entropy, before adding x. Assuming that we have pre-computed/recorded this value, all the other summations only sum over words v for which $C(xv) > 0$, which, in many cases, is much smaller than the vocabulary size.

Many other clustering techniques [Brown *et al.*, 1992] attempt to maximize $\sum_{Y,Z} P(YZ)\log\frac{P(Y\mid Z)}{P(Z)}$, where the same clusters are used for both. The original speedup formula uses this version, and is much more difficult to minimize. Using different clusters for different positions not only leads to marginally lower entropy, but also leads to simpler clustering.