To appear in Computational Linguistics and Chinese Language Processing

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

Jianfeng Gao
Microsoft Research, China
Beijing, 100080, P.R.C
jfgao@microsoft.com

Joshua T. Goodman
Microsoft Research, Redmond
Washington 98052, USA
joshuago@microsoft.com

Jiangbo Miao[1]
Department of Computer & Information Sciences
University of Delaware, USA

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 up to more than 40% size reduction at the same perplexity

1. Introduction

Statistical language modelling (SLM) has been successfully applied to 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 thus augment 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 reduced 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 previous research on word clustering has focused on how to get the best clusters. We have

[1] This work was done while the author was visiting Microsoft Research China.

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 review briefly 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 to find the 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:

$$P(w_n) = P(w_n \mid w_1 \text{ L } w_{n-1}) \tag{1}$$

In practice, the cases of *n*-gram models that people usually use are for *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:

$$P(w_n \mid w_1 \text{ L } w_{n-1}) \approx P(w_n \mid w_{n-2} w_{n-1}) \tag{2}$$

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 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:

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

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 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:

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

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 present 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. Their 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 is also 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), and "Sunday" (星期天) etc.

In data-driven clustering, words are clustered automatically in a 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 paper, we focus our research on novel techniques of using clusters rather than different ways of finding clusters. We also notice that all realistic applications have memory constraints. Therefore, we concentrate 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. As will be shown in Section 5, most experimental results are presented in the form of size/perplexity curves.

3. Using Clusters

In this section, we describe our techniques of 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*, w_1 and w_2 are context words to predict w_3, called the *conditional word*. Either the predicted word or the conditional word can be clustered in 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 have 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:

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

Now, since each word belongs to a single cluster, $P(c_i|w_i)$=1, and thus

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

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 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*" and thus, the probability *P*(*WEEKDAY* | *party on*) will be relatively high. And 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 could 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

$$P(Tuesday \mid party\ on) = P(WEEKDAY \mid EVENT\ PREPOSITION) \times P(Tuesday \mid EVENT\ PREPOSITION\ WEEKDAY)$$

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; and is 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

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

The approximation is suboptimal unless we use high (count) cutoffs for bigram and trigram. Given that combined clustering uses more information than regular IBM clustering, we assume 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 presented, but previous studies showed that no one outperformed others significantly. In this paper, we do not explore different techniques for finding cluster, but simply pick one we think 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. But, there are very few words that can follow both "*a*" and "*an*", and so, for conditional cluster, 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 experiment, 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 the conditional clusters, we try to minimize the perplexity of 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 the predictive clusters, we try to minimize the perplexity of 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 is 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 the 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 report our main experiments. In Section 5.1, we describe the text corpus we used. In Section 5.2, we compare the performance of word-based trigram models with cluster-based *n*-gram models. We show perplexity results of cluster-based *n*-gram models alone, as well as combined models where the cluster-based *n*-gram models were interpolated with word-based *n*-gram models. In Section 5.3, we present a fairly thorough comparison of different techniques of using clusters for language model compression. We then show that our novel clustering techniques can produce much smaller models at a given 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 set. 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 is of great variety in domain as well as in style. It is 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 is 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 are fairly open tests, since we used heterogeneous (in domain and style) data sets from different sources for language model training and testing. Thus, we assume that problems from data sparseness and training-test mismatch are 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 set 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 are more like closed tests, since we used homogeneous (at least in style) data sets from the same corpus for language model training and testing. We then assume that data sparseness and training-test mismatch are less serious than for the Chinese corpus. We also assume that the Japanese lexicon is 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).

By using the abovementioned Chinese and Japanese text corpora, we would like 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 of 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 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 8^{th} level, there will be roughly 2^8=256 clusters. Since the tree is not balanced, the actual number of clusters may be somewhat smaller. So in what follows, we 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 Chinese corpus (Zhou, 1996), (2) the 1428 semantic clusters of the Chinese corpus, which is taken from "同义词词林" (TongYiCi CiLing), a widely used Chinese thesaurus (Mei, 1983). As shown in Table 1, the perplexity is 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 one, but for larger sizes (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. It does not mean that human defined clusters are unreasonable or worse than automatically derived clusters. The two cluster sets are generated by different criteria and motivations. Hand clustering is usually based on semantic/syntactic similarity, while automatic clustering uses the perplexity measurement directly. So 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.

Number of clusters	Chinese	Japanese
2^5	644.31	346.05
2^6	542.13	268.92
2^7	464.76	223.70
2^8	405.92	194.26
2^9	358.57	172.63
2^10	322.13	155.39
28 (POS clusters)	1038.56	------
1428 (semantic clusters)	676.87	------
Word trigram	242.74	106.33

Table 1: Test set perplexities with cluster-based trigram models

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 outperform baseline models consistently. Unlike the case 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 perplexity reduction: about 3% on the Chinese corpus (at 2^9 clusters), and more than 10% on the Japanese corpus (at 2^9 clusters).

Number of clusters	Chinese	Japanese
2^5	236.01	100.01
2^6	235.02	98.21
2^7	234.21	96.68
2^8	233.53	95.73
2^9	233.42	95.41
2^10	234.11	95.66
2^11	234.81	96.72
2^12	235.53	97.60
2^13	236.58	99.58
Word trigram	242.74	106.33

Table 2: Test set perplexities with combined trigram models

5.2.3 Using high-order *n*-gram models

While the trigram approximation has been proven, in practice, to be reasonable, there is an argument that longer context can be helpful. This leads to research on using *n*-gram models where *n*>3, called higher-order *n*-grams. Most 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 other than smoothing. It is 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 on 2^8, and built a series of *n*-gram models, with *n* 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 into very high order *n*-gram models. After 10-gram models, depending on the corpus, we obtained approximately 10% perplexity reduction on 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).

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

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

5.2.4 Analysis of words in clusters

We divided the 50,180-entry Chinese lexicon into 2^8 clusters by automatic clustering. The number of words in each cluster is highly varied 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 occur less than 10 times in the training corpus. For each remaining cluster in table 4, we give top 15 to 30 two-character Chinese words with the highest frequency (at lest 10 times) in the training corpus.

We see that most of the words in each cluster belong to the same syntactic class, namely verbs for cluster **A**, nouns for cluster **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 of 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 game, 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 frequent words, look quite reasonable. It turns out that, given enough 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.

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 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), …

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

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. 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 take memory constraints into consideration, and concentrate 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 show that our novel clustering techniques can be combined with language model pruning methods to produce much smaller models at a given perplexity than 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, as well as 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, $\overline{P}(w|\overline{h})$,

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

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 is calculated for each *n*-gram entry, and we remove entries and reassign the deleted probability mass to backoff mass for any *n*-gram entry whose deletions increases the Kullback-Leibler distance by less than a specified threshold value. Compared to traditional count cutoff methods, Stolcke pruning performed a little better (Goodman and Gao, 2000). More importantly, the Stolcke method can prune a model to a specific size, simply by finding the threshold level that results in a model of that size. For all 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 comparing our different models. For these experiments, the baseline system is 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 have the best result at about 2^6 clusters for both the Chinese and Japanese corpora. Depending on the corpus, comparing to the baseline systems, at the same size, we get 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 get 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 have the best result at 2^6 clusters for both of them, the results at other numbers of clusters (e.g. 2^4, 2^7) are very different. For the Chinese corpus, all predictive clustering models performed about the same. For the Japanese corpus, models at larger number of clusters performed much better than models at small clusters (e.g. 2^4). In general with our clustering, when there is only a small amount of training data, the clusters become less useful. Perhaps, this is because there is a more serious data sparseness problem for the Chinese corpus, and

lots of parameters are out of training, thus larger clusters do not bring benefits. For the Japanese corpus, the data sparseness problem is much less serious, so a large number of clusters lead to significant perplexity reduction.

We also tried to set different pruning threshold values for the two components of predictive clustering models. We could not obtain any improvement. Therefore, in what follows, we always assume that we used the same pruning threshold value for both components in 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 two languages are qualitatively very similar. The performance is consistently improved by increasing the number of clusters. But no conditional clustering model beats 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.

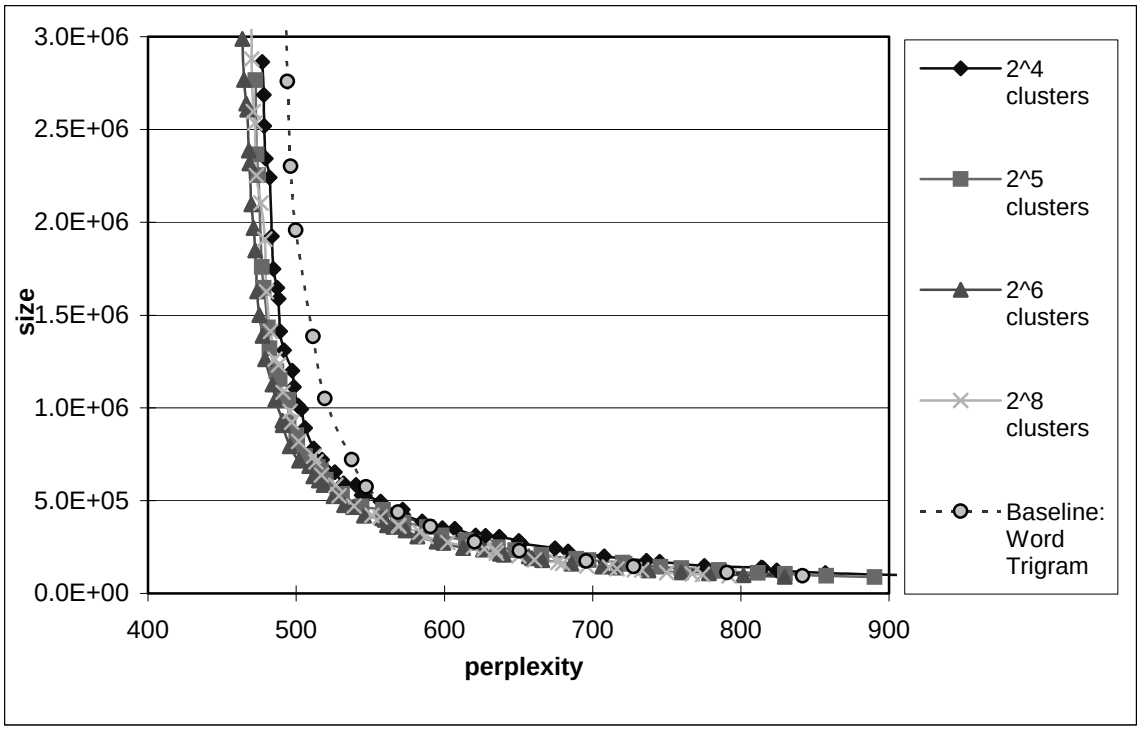

Figure 1. Comparison of predictive models with different number of clusters on Chinese corpus.

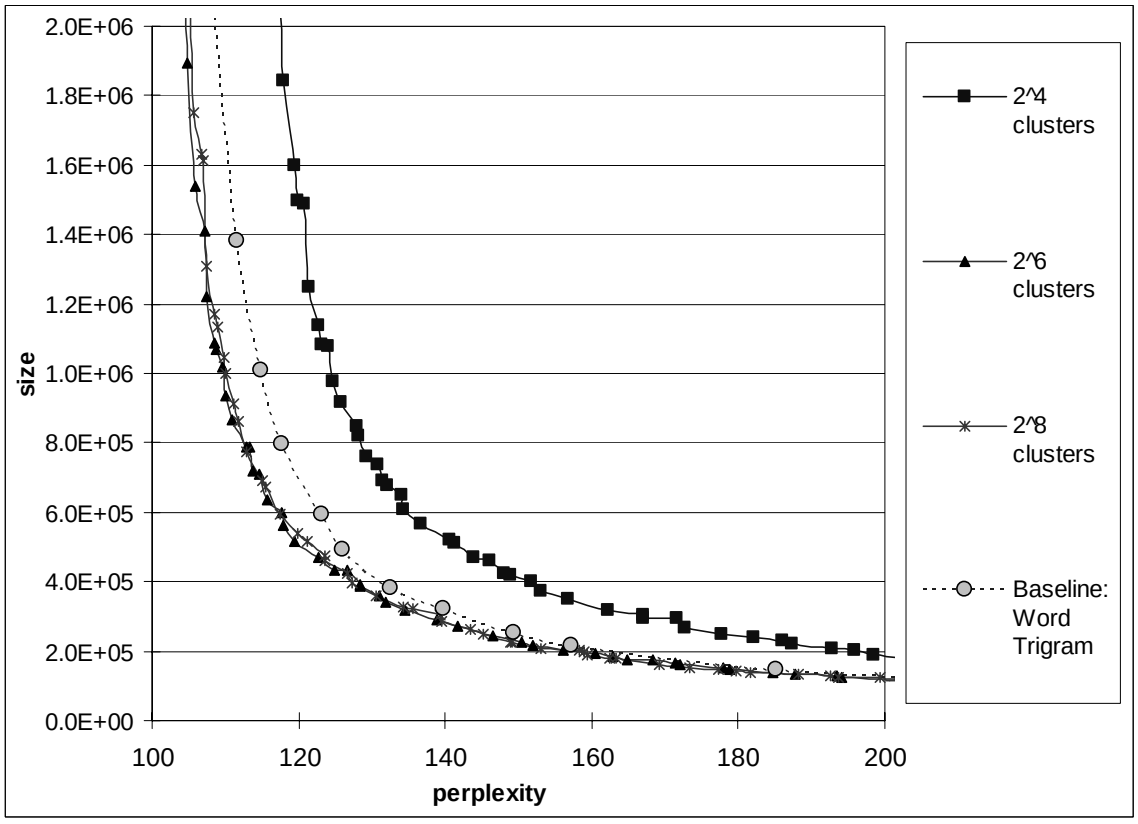

Figure 2. Comparison of predictive models with different number of clusters on Japanese corpus.

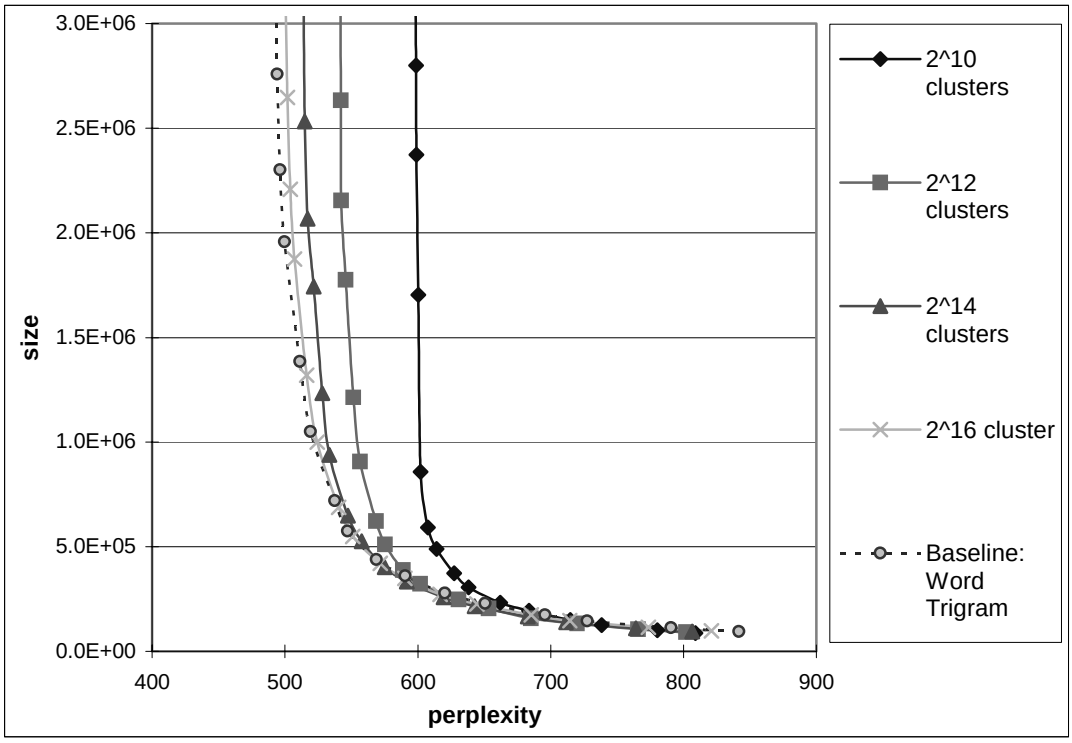

Figure 3. Conditional models with different number of clusters on Chinese corpus

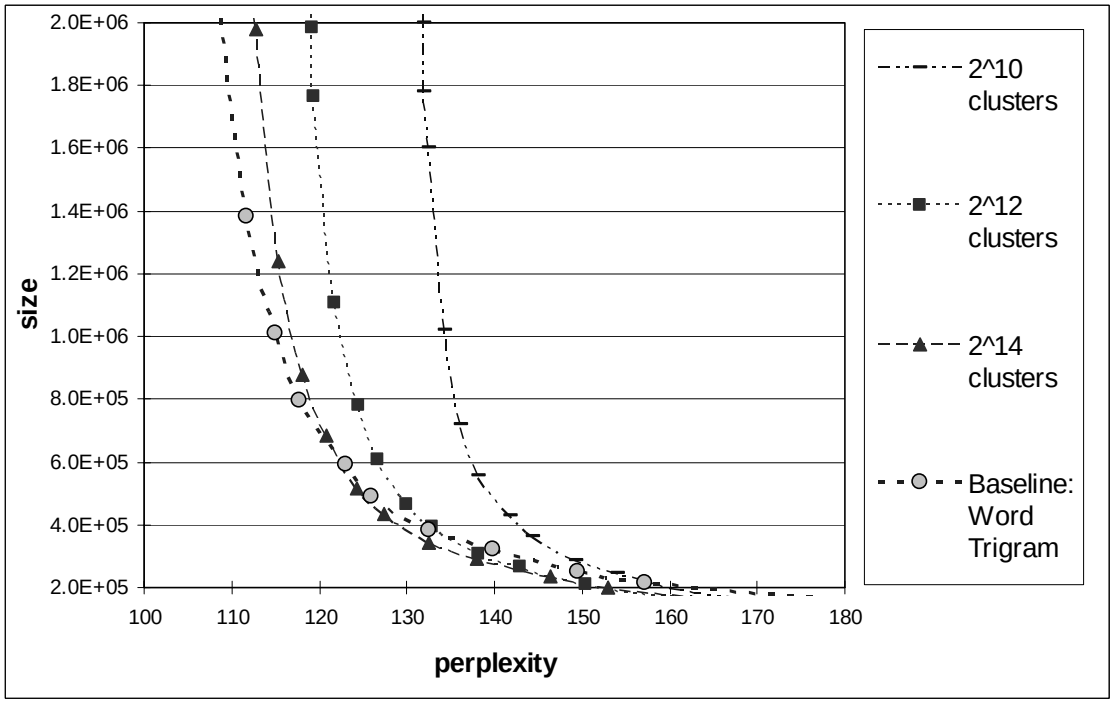

Figure 4. Comparison of conditional models with different number of clusters on Japanese corpus

5.3.3 Combined clustering

Now, we used combined clustering of Equation (12). As mentioned early, 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 present detailed analysis of parameter settings of the combined clustering model in (Goodman and Gao, 2000). In this paper, we only report 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 points of all 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 is 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 do not graph the first, worse point. We show the outer envelope of 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 like Figure 5. Instead, we show results of the top three best parameter settings we obtained, for instance, (2^4, 2^12) represent 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 have the best result at 2^14 conditional clusters and 2^6 predictive clusters. At large model sizes, we have the best result at 2^12 conditional clusters and 2^4 predictive clusters. We achieved a maximum 6.5% perplexity reduction at the same size, and a 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 at very large number 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 optimal to use conditional clusters no less than the vocabulary size, i.e. no conditional clustering.

Now, consider IBM clustering of Equation (6), which is a special case of combined clustering model. As shown in Figure 6, it is by far the worst, with roughly an order of magnitude worse performance than the others. 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 conditions the probability of the word also on the previous clusters. We compared this model to 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 outperform standard IBM models (e.g. we achieved 4% perplexity reduction at 2^9 clusters), and for the Japanese corpus, they work about the same. Notice that for these experiments, no pruning was done.

We summarize all experiments in this subsection in Figures 7 and 8. It turns out clearly that our novel clustering techniques produce much smaller models than previous methods (i.e. baseline systems) at the same perplexity level. In addition, several more detailed conclusions are suggested:

1. Conditional clustering does not help for both the Chinese and 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 outperform 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 outperforms the baseline system consistently. Although our results in this paper showed that

combined clustering achieved no improvements, in (Goodman and Gao, 2000), we showed that with more sophisticated techniques, it appears possible to make combined clustering better than predictive clustering.

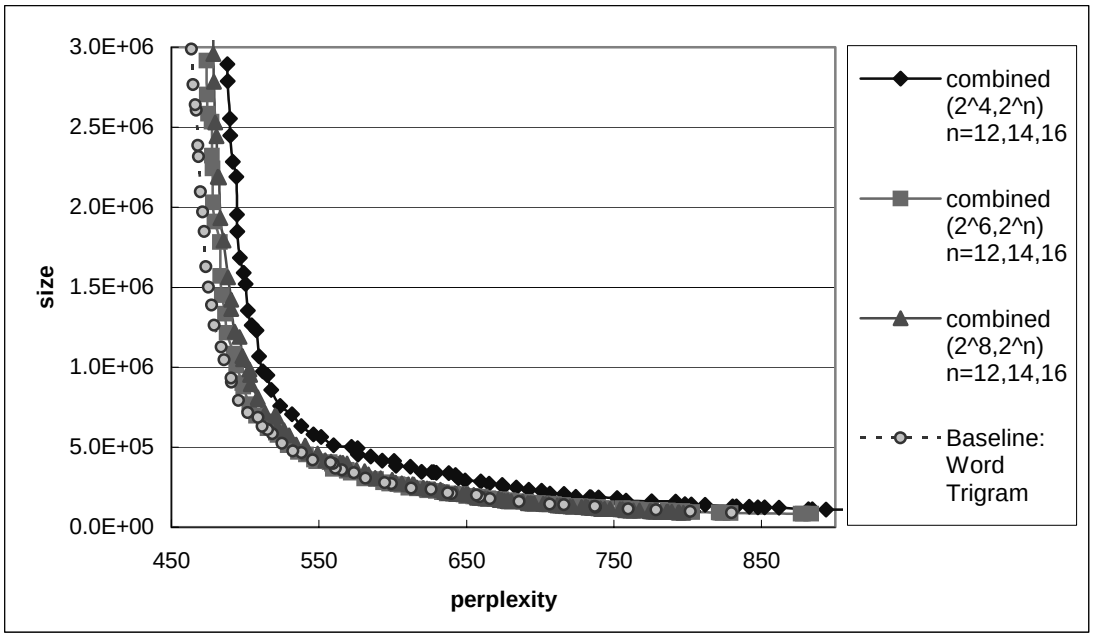

Figure 5. Comparison of combined clustering models with different number of clusters on Chinese corpus

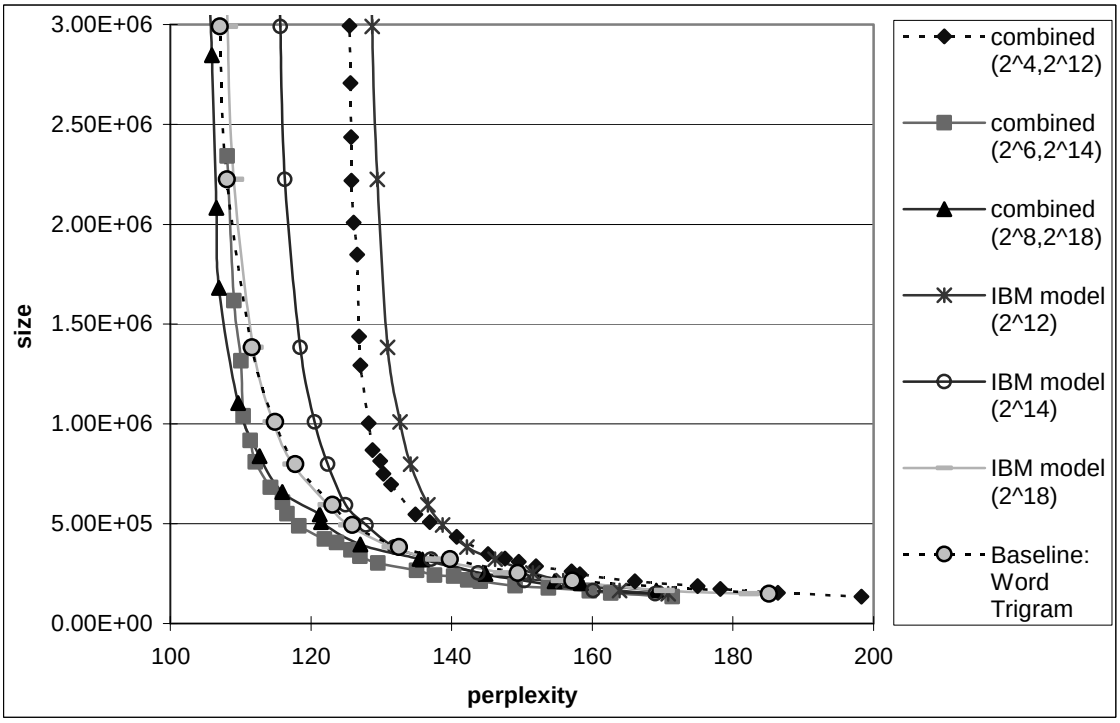

Figure 6. Comparison of combined clustering models with different number of clusters on Japanese corpus, and the IBM model.

Number of clusters	Equation (9)	Equation (20)
2^6	235.02	226.65
2^7	234.21	224.65
2^8	233.53	224.29
2^9	233.42	224.99
2^10	234.11	226.53
2^11	234.81	228.26
2^12	235.53	230.95
2^13	236.58	234.78
word trigram	242.74	242.74

Table 5: Comparison of different combined trigram models on Chinese corpus

Number of clusters	Equation (9)	Equation (20)
2^6	98.21	97.06
2^7	96.68	96.42
2^8	95.73	96.33
2^9	95.41	96.41
2^10	95.66	96.82
2^11	96.72	97.57
2^12	97.60	98.50
2^13	99.58	100.52
word trigram	106.33	106.33

Table 6: Comparison of different combined trigram models on Japanese corpus

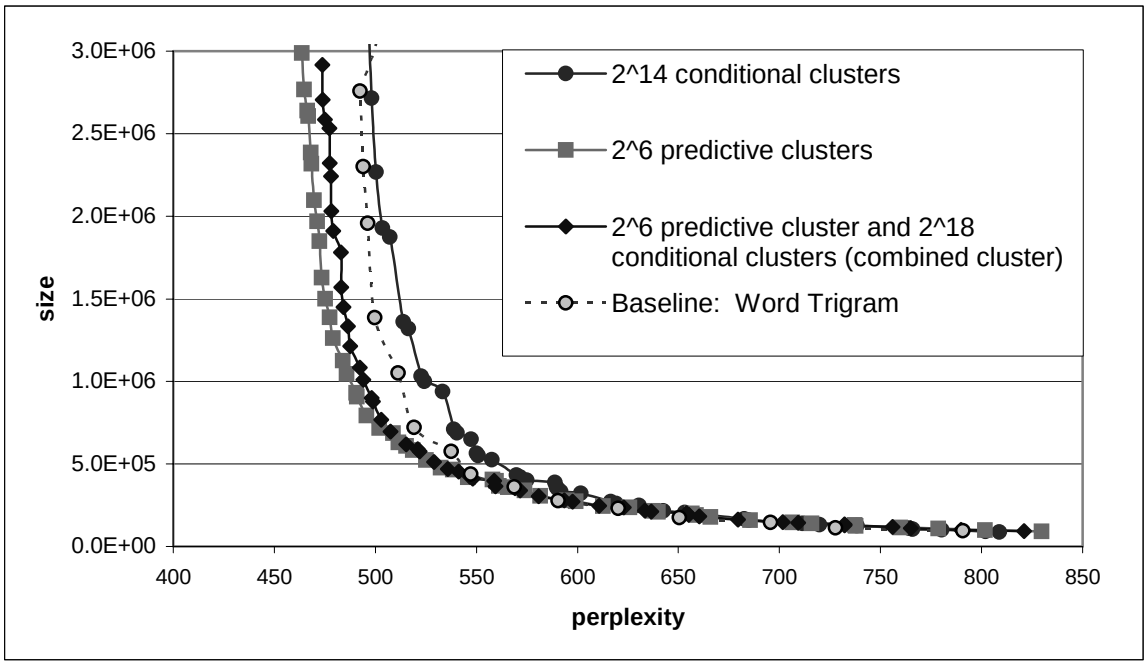

Figure 7. Summary of clustering models on Chinese corpus

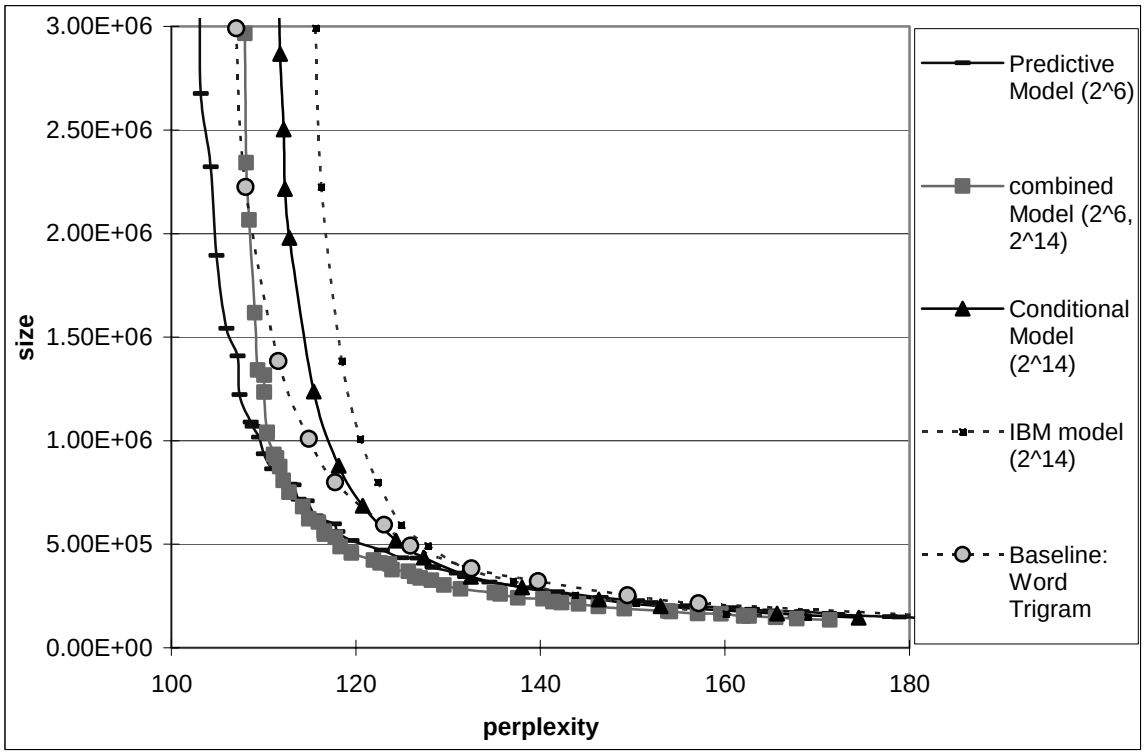

Figure 8. Summary of clustering models on Japanese corpus

6. Conclusion

Cluster-based *n*-gram models are a variation on traditional word-based *n*-gram models. They attempt to make use of the similarities between words. In this paper, we present an empirical study of clustering techniques for Asian language modeling. 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. We 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 work much better than previous methods. They not only showed better performance when interpolated with normal *n*-gram models, but can be combined with Stolcke pruning to produce models much smaller than unclustered models with the same perplexity.

Most language modeling improvements, reported previously, require significantly more space than the normal trigram baseline model, or have higher perplexity. Their practical value is questionable. In this paper, we proposed a technique that results in lower perplexity than traditional trigram models at every memory size. In other research (Gao et al., 2001) we have shown that cluster-based models of this form can be used effectively for 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 from Microsoft Research, Yoshiharu Sato, and Hiroaki Kanokogi from the Microsoft (Japan) IME group, for their help in developing the ideas and implementation in this paper. We would also like to thank Jiang Zhu, and Miyuki Seki, for their help in our experiments and providing Chinese and Japanese text corpora.

References

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

Bellegarda, J. R., Butzberger, J. W., Chow, Y. L., Coccaro, N. B., and Naik, D. (1996). A novel word clustering algorithm based on latent semantic analysis. In *ICASSP-96*, 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. (1990). A statistical approach to machine translation. *Computational Linguistics,* 16(2), pp. 79-85.

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

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

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

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

Gao, J., Goodman, J., Li, M., and Lee, K. F. (2001). Toward a unified approach to statistical language modeling for Chinese. To appear in *ACM Transactions on Asian Language Information Processing*.

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

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

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

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

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

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

Jelinek, F. (1990). 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. (2000). 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.

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

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

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

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

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

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

Miller, D., Leek, T., and Schwartz, R. M. (1999). 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. (1997). Evaluation of a language model using a clustered model backoff. In *ICASSP-97*, pp. 390-393.

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

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

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

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

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

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

Stolcke, A. (1998). 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. (1996). An extended clustering algorithm for statistical language models. *IEEE Transactions on Speech and Audio Processing*, 4(4): 313-316.

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

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

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

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

Zue, V. W. (1995). 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 describe methods for language model training. These include the modified absolute discounting smoothing method and Stolcke's entropy-based pruning method.

Absolute Discounting

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

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 use a slightly different technique in this research, modified absolute discounting. First, we describe basic absolute discounting. Letting D represent a discount, we set the probability as follows:

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

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

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

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, a different D is used for the bigram and trigram.

A further improvement is to use multiple discounts D. Taking the trigram as an example, D_1 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) introduce an estimate for the optimal D for absolute discounting smoothing as a function of training data counts[2]. In practice, we can use Equation (23) to Equation (26) to approximately estimate the optimal value for D_1, D_2, and D_3.

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

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

[2] Thanks to Ries, K.

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

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

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

Notice that for experiments in this paper, we do not use this approximation, but instead optimize the discounts on heldout data. This leads to very limited improvements.

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), $\alpha'(w_{i-2}w_{i-1})$ 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 a 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, assume that the model we are using is of the form $P(z|Y)$; we want to find the placement of words *y* into clusters *Y* that minimizes 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 for speeding up clustering was to use a top-down approach. We note that agglomerative clustering algorithms – those which merge words bottom up – may require significantly more time than top-down, splitting algorithms. Thus, our basic algorithm is top-down. However, at the end, we sometimes perform four iterations of swapping all words between all clusters. Notice that for experiments reported in this paper, we used the basic top-down algorithm without swapping.

Another technique we use 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. So, we proceed top down, splitting clusters. When we are ready to split a cluster, we randomly pick a few words, and put them into two random clusters, and then swap them in such a way that entropy is decreased, until convergence

(no more decrease can be found). Then we add a few more words, typically $\sqrt{2}$ more, and put each into the best bucket, then swap again until convergence. This is repeated until all words in the current cluster have been added and split. We haven't tested this particularly thoroughly but out intuition is that it should lead to large speedups.

We use one more important technique that speeds computations, adapted from earlier work of (Brown et al., 1992). We attempt to minimize the entropy of our clusters. Let *v* represent words in the vocabulary, and *W* represent a potential cluster. We minimize

$$\sum_{v} C(Wv) \log P(v \mid W)$$

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,

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

The first summation in Equation 28 can be computed relatively efficiently, in 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

$$\sum_{v|C(xv)=0} C(Xv)\log P(v \mid X)$$
$$= \sum_{v} C(Wv)\log P(v \mid W) - \sum_{v|C(xv)\neq 0} C(Wv)\log P(v \mid W) + \left(\log \frac{C(W)}{C(X)}\right)\left(C(W) - \sum_{v|C(xv)\neq 0} C(Wv)\right) \quad (32)$$

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 complex to minimize, Using different clusters for different positions not only leads to marginally lower entropy, but also leads to simpler clustering.