Toward a Unified Approach to Statistical Language Modeling for Chinese

JIANFENG GAO
JOSHUA GOODMAN
MINGJING LI
KAI-FU LEE
Microsoft Research

This article presents a unified approach to Chinese statistical language modeling (SLM). Applying SLM techniques like trigram language models to Chinese is challenging because (1) there is no standard definition of words in Chinese; (2) word boundaries are not marked by spaces; and (3) there is a dearth of training data. Our unified approach automatically and consistently gathers a high-quality training data set from the Web, creates a high-quality lexicon, segments the training data using this lexicon, and compresses the language model, all by using the maximum likelihood principle, which is consistent with trigram model training. We show that each of the methods leads to improvements over standard SLM, and that the combined method yields the best pinyin conversion result reported.

Categories and Subject Descriptors: I.2.7 [**Artificial Intelligence**] Natural Language Processing – *speech recognition and synthesis*; H.1.2 [**Models and Principles**]: User/Machine Systems – *Human information processing*; H.5.2. [**Information Interfaces and Presentation**]: User Interfaces – *Natural language*

General Terms: Experimentation, Human Factors, Languages, Measurement

Additional Key Words and Phrases: Statistical language modeling, n-gram model, smoothing, backoff, Chinese language, lexicon, word segmentation, domain adaptation, pruning, Chinese pinyin-to-character conversion, perplexity, character error rate

1. INTRODUCTION

Statistical language modeling (SLM) has been successfully applied to many domains such as speech recognition [Jelinek 1990], information retrieval [Miller et al. 1999], and spoken language understanding [Zue 1995]. In particular, trigram models have been demonstrated to be highly effective for these domains. In this article we extend trigram modeling to Chinese by proposing a unified approach to SLM.

Chinese has some special attributes and challenges. First, there is no standard definition of a word, and there are no spaces between characters. But statistical language models require word boundaries. Second, linguistic data resources are not yet plentiful in China, so the best source of training data may be the Web. However, the quality of data from the Web is questionable. To address these two issues, we ideally need a system that can automatically select words from the lexicon, segment a sentence into words, filter

Authors' addresses: Jianfeng Gao, Mingjing Li, Kai-Fu Lee, Microsoft Research (Asia), Zhichun Rd 49, Beijing, 100080, China; emails: jfgao@microsoft.com; mjli@microsoft.com; kfl@microsoft.com. Joshua Goodman, Microsoft Research (Redmond), Washington 98052; email: joshuago@microsoft.com.

high-quality data, and combine all of the above in an SLM that is memory-efficient. Extending our previous work in Gao et al. [2000b], this article presents a unified approach to solving these problems by extending the maximum likelihood principle in trigram parameter estimation. We introduce a new method for generating lexicons, a new algorithm for segmenting words, a new method for optimizing training data, and a new method for reducing language model size. All these methods use a perplexity-based metric, so that the maximum likelihood principle is preserved.

This article is structured as follows: In the remainder of this section we present an introduction to SLM, *n*-gram models, smoothing, and performance evaluation. In Section 2 we give more details about processing Chinese and present the overall framework. In Section 3 we describe a new method for jointly optimizing the lexicon and segmentation. In Section 4 we present a new algorithm for optimizing the training data. In Section 5 we give our method for reducing the size of the language model. In Section 6 we present the results of our main experiments. Finally, we conclude in Section 7.

1.1 Language Modeling and *N*-gram Models

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

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

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

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

An estimate for the probability $P(w_i \mid w_{i-2} w_{i-1})$, given by Eq. (3), is 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, making it impossible for a system like speech recognition to accept a 0 probability sequence like this. So these probabilities are typically smoothed [Chen and Goodman 1999]: some probability is removed from all nonzero counts and is used to add probability to the 0 count items. The added probability is typically in proportion to some less specific, but less noisy, model. Recall that for language modeling, a formula of the following form is typically used:

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

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 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, the 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.

In this article a character perplexity PP_C, especially defined for the Chinese language, is also used. The definition is similar to PP_W, as follows:

$$PP_C = 2^{-\frac{1}{N_C}\sum_{i=1}^{N_W}\log_2 P(w_i|w_{i-2}w_{i-1})} \quad (6)$$

where N_C is the total number of characters in the test set. Note that both PP_C and PP_W are based on the word trigram model probability $P(w_i|w_{i-2}\ w_{i-1})$, so PP_C is related to PP_W by the following equation,

$$PP_C = PP_W^{\frac{N_W}{N_C}} \ . \quad (7)$$

An alternative, but equivalent, measure to perplexity is *cross-entropy*, which is simply $\log_2$ of perplexity. This value can be interpreted as the average number of bits needed to encode the test data using an optimal coder. We sometimes refer to cross-entropy as simply entropy.

For applications such as speech recognition, handwriting recognition, and spelling correction, it is generally assumed that lower perplexity/entropy correlates with better performance. In Section 6.5.3 we present results that indicate this correlation is especially strong when the language model is used for the application of pinyin to Chinese character conversion, which is a similar problem to speech recognition.

In this article we use the perplexity measurement due to its pervasive use in the literature.

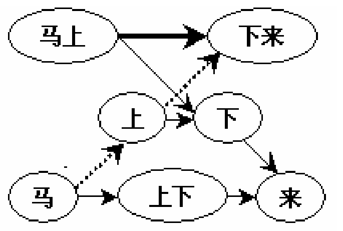

Fig. 1. The word graph of the Chinese sentence “马上下来”

2. A UNIFIED APPROACH TO CHINESE STATISTICAL LANGUAGE MODELING

In this section we give more details about processing Chinese. We describe previous Chinese trigram language model training and present some open issues. To address these issues, we present our overall framework: a unified Chinese statistical language modeling approach. This approach is something like an ideal concept, and is by no means fully developed; but it drives almost all ongoing research on Chinese statistical language modeling at Microsoft Research Asia.

2.1 The Chinese Language

The Chinese language is based on characters. There are 6763 frequently used Chinese characters. Each Chinese word is a semantic concept that is about 1.6 characters on average. But there is no standard lexicon of words–-linguists may agree on some tens of thousands of words, but they will dispute tens of thousands of others. Furthermore, Chinese sentences are written without spaces between words, so a sequence of characters will have many possible parses in the word segmentation stage. Figure 1 shows the segmentation of a simple sentence with only four characters. The four characters can be parsed into words in five ways. For example, the dotted path represents “*dismounted a horse,*” and the path in boldface represents “*immediately coming down.*” This figure also shows seven possible “words” about some of which (e.g., 上下) there might be some dispute as to whether they should be considered “words” at all.

2.2 Chinese Trigram Language Model Training

Due to the problems mentioned above, although word-based approaches (e.g., word-based language models) work very well for Western languages where words are well defined, they are difficult to apply to Chinese. We might think that character language models could bypass the issue of word boundaries, but previous work [Yang et al. 1998] found that a Chinese SLM built on characters did not yield good results. So our approach should be word-based, and thus requires a lexicon and a segmentation algorithm.

Another problem related to SLM, and particularly to Chinese SLM, is the collection of a good training data set. This is particularly relevant for Chinese, because the organization of linguistic data resources is just starting in China. We solve this problem by using data from the Web, a technique that can be relevant to any language because the Web is growing at much faster pace than any linguistic data resource. Unfortunately,

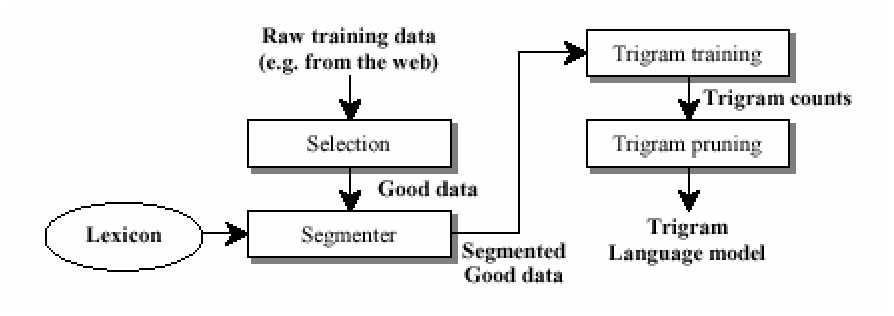

Fig. 2. Trigram model training for the Chinese language.

the quality of Web data is highly variable, so it becomes very important to be selective, to filter large amounts of data, and to select the portions that are suitable.

A flowchart of typical Chinese language model training is illustrated in Figure 2. It consists of several serial steps: after being collected (e.g., from the Website), the training text is segmented on the basis of a predefined lexicon. The trigram language model is then trained on the segmented training set. Finally, the model is pruned to meet the memory limits of the application.

This serial, straightforward Chinese language model training has the following problems:

1. Selecting an optimal training set from raw data is a very expensive and tedious task, whereas automatic selection remains an open issue.
2. The definition of the lexicon is made arbitrarily by hand, and is not optimized for language modeling.
3. Segmentation is usually carried out by a greedy algorithm (e.g., maximum matching), which does not integrate with other steps, and is not optimal.
4. Trigram training is based on the lexicon and the segmented training data set. However, as mentioned above, decisions about the lexicon, segmentation, and training set are made separately, and are not optimized for trigram training. Thus, the resulting trigram model is suboptimal.
5. Count cut-offs [Jelinek 1990], which are widely used to prune the trigram, are not sensitive to the performance of the model (i.e., perplexity).

2.3 The Unified Chinese Statistical Language Modeling Approach

To address the problems mentioned above, in this article we present a unified approach that extends the maximum likelihood principle used in trigram parameter estimation to the problems of selecting the lexicon, the training data, and word segmentation. In other words, we want to: *select the training data subset (adapt it to a specific domain if necessary), select a lexicon, and segment the training data set using this lexicon, all in a way that maximizes the resulting probability (or reduces the resulting perplexity) of the training set.*

In formulating this problem we also realized that this optimization should not be limitless, since all applications have memory constraints. So the above questions should be asked subject to memory constraints, which could be arbitrarily large or small. Conceptually, we would like to arrive at the architecture shown in Figure 3: given an

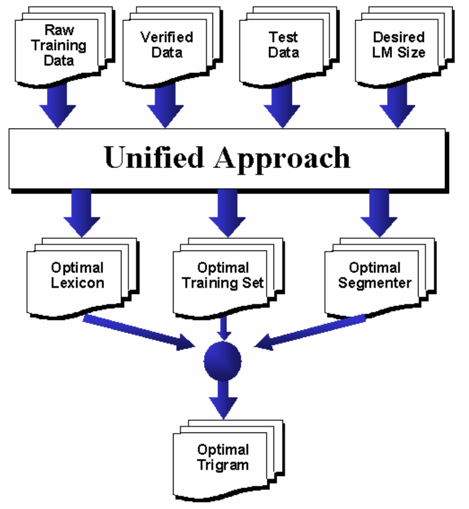

Fig. 3. The unified language modeling approach.

application's independent open test set, a large training set (e.g., raw data from the Web), a small verified data set (e.g., available application documents), and a maximum memory requirement, we optimize the lexicon, word segmentation, and training set, resulting in an optimal trigram model for the application.

In the next section we describe some of the ongoing projects of this unified approach, including (1) lexicon and segmentation optimization; (2) training set optimization; and (3) language model pruning. All use the maximum likelihood principle, i.e., to minimize the perplexity of the resulting language model.

3. OPTIMIZING THE LEXICON AND SEGMENTATION

This section addresses optimizing lexicon selection and corpus segmentation. We first describe a simple method for constructing a lexicon from a very large corpus. Next, we describe an algorithm for the joint optimization of the lexicon, segmentation, and language model.

Previous systems [Yang et al. 1998; Wong et al. 1996] usually make *a priori* decisions about the lexicon as well as segmentation, and then train a word trigram model. Instead, in this article we treat the decision of lexicon and word segmentation as a hidden process for Chinese SLM. Thus, we could use the powerful *expectation maximization* (EM) algorithm to jointly optimize the hidden process and the language model.

3.1 Lexicon Construction from Corpus

In traditional rule-based approaches, much human effort is required to extract words/compounds automatically from a large corpus; but statistical approaches have recently come into wide use. In Yang et al. [1998], the elements of the lexicon can be any "segment patterns" extracted from the training corpus with the goal of minimizing the overall perplexity. The same perplexity-based metric is also used by Giachin [1995] and

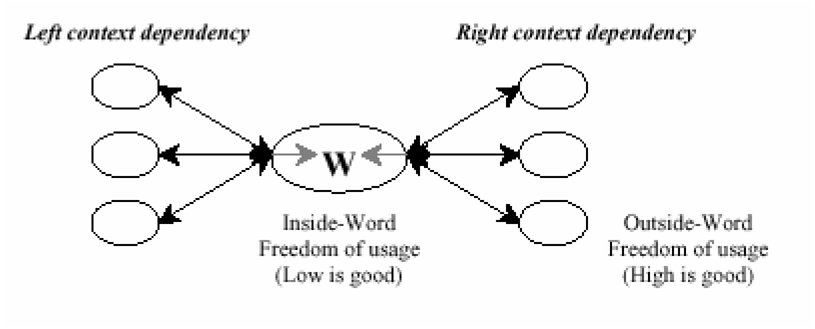
Left context dependency
Right context dependency
W
Inside-Word Freedom of usage (Low is good)
Outside-Word Freedom of usage (High is good)

Fig. 4. The mutual information and context dependency of a word.

Berton et al. [1996] to add and remove lexicon items. However, in practice, finding an optimal lexicon on the basis of perplexity estimates is very computationally expensive. Hence, approximate approaches are used where words and compounds are extracted via statistical features, since these are easier to obtain. For example, Chien [1997] proposed an approach based on the PAT-Tree to automatically extract domain-specific terms from online text collections. Chien used two statistical features, *associate norm* and *context dependency*. Similar examples (they only vary in different statistical feature sets) include Tung et al. [1994]; Wu et al. [1993]; and Fung [1998]. These methods achieved medium performance (i.e., word precision/recall) on relatively small corpora. But it is not clear whether these methods work properly with large corpora and in SLM for Chinese.

In this section we propose an efficient method for constructing a lexicon for Chinese SLM. We use an approximate information gain-like metric, consisting of three statistical features, namely (1) *mutual information*, (2) *context dependency*, and (3) *relative frequency*. The basic idea is that a Chinese word should appear as a stable sequence in the corpus. That is, the components within the word should be strongly correlated, while the components at both ends should have low correlations with outer words. This is illustrated in Figure 4.

Mutual information (MI) is a criterion for evaluating the correlation of different components (e.g., characters or short words) in the word. For example, let *MI(x,y)* denote the mutual information of a component pair *(x, y)*. The higher the value of *MI*, the more likely *x* and *y* are to form a word. The extracted words should have a higher *MI* value than a preset threshold. In Section 6.3 we examine the effect of different forms of mutual information estimates.

Context dependency (CD) is a criterion for evaluating the correlation of the candidate word and components outside at both ends. A character string *X* has left context dependency if

$$LSize = |L| < t_{size} \qquad (8)$$

or

$$MaxL = MAX_{\alpha} \frac{f(\alpha X)}{f(X)} > t_{freq} \qquad (9)$$

where t_{size}, t_{freq} are threshold values, *f(.)* is frequency, *L* is the set of left adjacent strings of *X*, $\alpha \in$, and $|L|$ is the number of unique left adjacent strings. Similarly, a character string

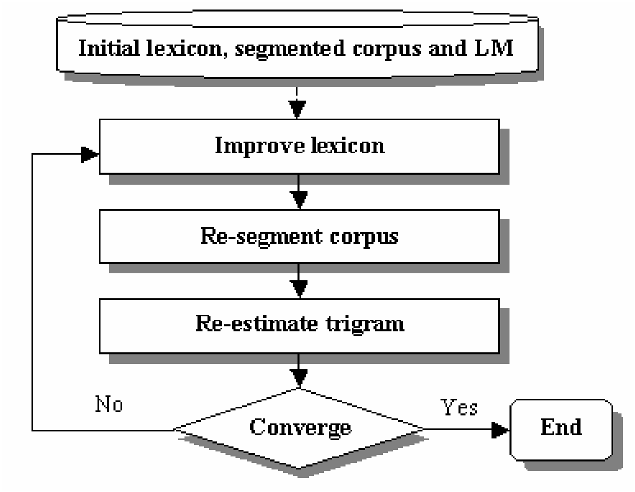

Fig. 5. The flowchart of the iterative method for lexicon, segmentation, and language model joint optimization.

X has right context dependency if

$$RSize = | R | < t_{size} \tag{10}$$

or

$$MaxR = MAX_{\beta} \frac{f(X\beta)}{f(X)} > t_{freq} \tag{11}$$

where t_{size}, t_{freq} are threshold values; *f(.)* is frequency; *R* is the set of right adjacent strings of *X*; and $\beta \in R$ and $|R|$ are the number of unique right adjacent strings. The extracted word should have neither left nor right *context dependency*.

Relative frequency (RF) is a criterion to reduce noise in the lexicon. All words with lower frequency are removed from the lexicon.

The threshold values of the metric (i.e., MI, CD, and RF) are defined empirically in our experiments, as described in Section 6.2.1.

3.2 Joint Optimization of Lexicon and Segmentation

Previous research [Yang 1998] has shown that separate optimizations of the lexicon and segmentation can lead to improved results. We propose a new iterative method for joint optimization of the lexicon, segmentation, and language model. This method aims to minimize perplexity, so that it is consistent with the EM criterion. There are four steps in this algorithm: (1) initialize, (2) improve lexicon, (3) resegment corpus, and (4) reestimate trigram. Steps 2 through 4 are iterated until the overall system converges. This algorithm is shown in Figure 5.

3.2.1 Initialization

We can obtain the initial lexicon by automatically extracting words/compounds from a corpus by using statistical features, as described in Section 3.1. An alternative method for obtaining the initial lexicon is to take the intersection of several humanly compiled lexicons, with the assumption that if all lexicographers include a word, then it is necessary to include it. We then use this lexicon to segment the corpus using a maximum matching algorithm [Wong and Chan 1996]. From this segmented corpus of word tokens, we computed an initial trigram language model.

3.2.2 Iterative Joint Optimization

We iteratively optimize the lexicon, segmentation, and the language model:

(1) Improve lexicon (lexicon optimization). From the segmented list, we obtain a "candidate list" of words to be added to the lexicon (we use a PAT-Tree-based approach similar to Chien's [1997] to create this candidate list). We then remove those words from the existing lexicon whose removal impacts perplexity least negatively, and then add to the lexicon those words from the "candidate list" whose addition most positively impacts perplexity. In our experiments, described in Section 6.2, we used the information gain-like metric, as described in Section 3.1.

(2) Resegment corpus. Given a Chinese sentence, which is a sequence of characters, $c_{1,}\ c_2 ... c_n$, there are M (M>=1) possible ways to segment it into words. We can compute the probability $P(S_i\ )$ of each segmentation S_i based on the trigram language model. Then, $S_{k=argmax}\ P(S_i\ )$ is selected as the correct one. The Viterbi search is used to find S_k efficiently.

(3) Reestimate trigram. We reestimate the trigram parameters, since by this time the lexicon and the segmentation have changed.

4. OPTIMIZING THE TRAINING SET

In applying an SLM, it is usually the case that more training data will improve a language model. However, blindly adding training data can cause several problems. First, if we want to use data of variable quality (from the Web, for instance) adding data (for example, data with errors) could actually hurt system performance. Second, even if we filter good data, we may want to balance it among all the training data, in order to give greater emphasis to data that better matches real usage scenarios or better balances our overall training set. Finally, there is never infinite memory, and every application has a memory limit on the size of the language model.

Our approach here is to take a small set of high-quality corpora (e.g., available application documents), called the *seed set*, and a large but mixed-quality corpus (e.g., data collected from the Web), called the *training set*, and train a language model that not only satisfies the memory constraint but also has the best performance.

In this section we describe two methods for optimizing a training set: one for filtering training data and the other for adapting training data.

4.1 Filtering the Training Set

To filter large amounts of data (e.g., data with errors) and select portions that are suitable for language modeling, we propose, subject to memory requirements, a new method to jointly optimize performance. The basic method has four steps: (1) segmenting training data; (2) ranking training units; (3) selecting and combining training data; and (4) pruning language models. Steps (3) and (4) are repeated until the improvement in the perplexity of the language model is less than a preset threshold.

4.1.1 Segmenting Training Data

The first step is to take the large training set and divide it up into units, so that we can decide whether to keep each unit and how much to trust each unit.

Expanding the idea of *TextTiling* [Hearst 1997], we propose an algorithm to automatically segment the training data into *N* units, satisfying a size-range constraint while maximizing similarity within units and maximizing differences between units. It involves the following steps:

1. Search for available sentence boundaries and empirically cluster approximately every 300 content words into a *training chunk*. We refer to the points between training chunks as *gaps*.
2. Compute the *cohesion score* at each gap. The cohesion score is the measure of the similarity between *training blocks* (a sequence of training chunks) on both sides of the gap. Due to the limited data within each unit, our score is based on smoothed within-block term frequency (TF). Formally, the score between two training blocks, b_1 and b_2, is the number of terms in common in both blocks.

$$Score\ (b_1, b_2) = \sum_W I(w_i = w_j, w_i \in b_1, w_j \in b_2)$$

   where I is an indicator function such that I_A=1 if A is true, and 0 otherwise, W is the vocabulary.
3. Select the *N-1* gaps with lowest cohesion scores. Each gap separates two units, and each unit has one or more chunks.

We also add a size-range constraint to avoid training units that are too small or too large.

4.1.2 Ranking Training Data

The second step is to assign a score to each unit. Following our unified approach, we use perplexity as our metric [Lin et al. 1997]. We train a language model from our seed set and measure each training data unit's test-set perplexity against this language model. Here we use a bigram model, since our seed set is not large enough to train a reliable trigram.

We then iteratively increase the seed model by adding *blind feedback* [Rocchio 1971], which is widely used for query expansion in information retrieval. Similar to the case of information retrieval, the basic idea is that if we trust the performance of the test-set perplexity measurement, the top-ranked training units may be considered as a similar training unit set to the seed set, and can be used as a seed set as well. In practice, we augment the initial seed set with training units in the top 5-8% of *N* training units and then retrain the seed language model. This process is iterated until the resulting seed set is sufficient to train a robust language model.

4.1.3 Combining Training Data

There are several ways to combine the selected training data with the seed set. We first combined them by simply adding the training units to the seed set.

But we found that better results could be obtained by interpolating the language model. Our language model interpolation algorithm involves: (1) clustering training units into *N* clusters; (2) training an *n*-gram back-off language model per cluster; and (3) interpolating all such language models into one by simple interpolation of the following form

$$P(w) = \sum_{i=1}^{N} \alpha_i P_i(w) \tag{12}$$

where α_i is the interpolation weight of the ith model, and $\sum_{i=1}^{N} \alpha_i = 1$. The interpolation weights are estimated by using the EM algorithm.

4.1.4 Pruning the Language Model

The widely used count cut-offs prune the language model by discarding *n*-gram counts below a certain cut-off threshold. It is, unfortunately, impossible to prune a language model to a specific size. Furthermore, in case of a combined language model, as described above, it is not known which of the original background probabilities will be useful in the combined model, so we cannot use count cut-offs.

Given a memory constraint, our system can produce a language model. We apply a relative entropy-based cut-off method [Stolcke 1998]. The basic idea is to remove as many "useless" probabilities as possible without increasing perplexity. This is achieved by examining the weighted relative entropy or Kullback-Leibler distance between each probability $P(w|h)$ and its value $\overline{P}(w|\overline{h})$ from the back-off distribution:

$$D(P(w \mid h) \,||\, \overline{P}(w \mid \overline{h})) = P(w \mid h) * \log \frac{P(w \mid h)}{\overline{P}(w \mid \overline{h})} \tag{13}$$

where $\overline{h}$ is the reduced history. When the Kullback-Leibler distance is small, the back-off 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 iteratively remove entries and reassign the deleted probability mass to back-off mass, until the desired memory size is reached.

In Section 5, we discuss our pruning method in more detail, extending the relative entropy-based method to a novel technique that also uses word clustering. In Section 6, we give experimental results, showing that this new technique outperforms traditional pruning methods.

4.2 Adapting a Training Set Domain

For specific domains, language modeling usually suffers from sparse-data problems. To remedy these problems, previous systems mixed language models built separately for specific and general domains [Iyer and Ostendorf 1997; Clarkson and Robinson 1997; Seymore and Rosenfeld 1997; Gao et al. 2000a]. The interpolation weight used to combine the models is optimized so as to minimize perplexity. However, in the case of combined language models, perplexity has been shown to correlate poorly with recognition performance, i.e., word error rate.

We find that *n-gram distribution characterizes domain-specific training data*. In this article we propose an approach based on adapting *n*-gram distribution for language model training, where we adapt the language model to the domain by adjusting the *n*-gram distribution in the training set to that in the seed set.

Instead of combining trigram models built on the training set and seed set, respectively, we directly combine trigram counts *C(xyz)* with an adaptation weight *W(xyz)* of the form

$$C(xyz) = \sum_i W_i(xyz) \times C_i(xyz) \tag{14}$$

where $W_i(xyz)$ is the adaptation weight of the *i*th training set estimated by

$$W_i(xyz) = \log \left(\frac{P(xyz)}{P_i(xyz)} \right)^{\alpha} \tag{15}$$

where α is the adaptation coefficient, *P(xyz)* is the probability of the trigram *(xyz)* in the seed set, and $P_i(xyz)$ is the probability of the trigram *(xyz)* in the *i*th training set. It is estimated by

$$P_i(xyz) = \frac{C_i(xyz)}{\sum_{xyz} C_i(xyz)} \tag{16}$$

The key issues in adapting the *n*-gram distribution are determining α and selecting the seed set, described in Section 6.3.3.

5. REDUCING LANGUAGE MODEL SIZE

Language models for applications such as large vocabulary speech recognizers are usually trained on hundreds of millions or billions of words. Typically, an uncompressed language model is comparable in size to the data on which it is trained. Some form of size reduction is therefore critical for any practical application. Many different approaches have been suggested for reducing the size of language models, including count cutoffs [Jelinek 1990]; weighted difference pruning [Seymore and Rosenfeld 1996]; Stolcke pruning [Stolcke 1998]; and clustering [Brown et al. 1990].

In this section, after a brief survey of previous work, we present a new technique that combines a novel form of clustering with Stolcke pruning.

In Section 6.4, we first present a comparison of these various techniques and then demonstrate that our technique performs better than a factor of 2 or more than Stolcke pruning alone. On our Chinese dataset, the performance improvement is at least 35% at all but very high perplexities.

None of the techniques we consider are lossless. Therefore, whenever we compare techniques, we do so by comparing the size reduction of the techniques at the same perplexity. We begin by comparing count-cutoffs, weighted difference pruning, Stolcke pruning, and variations on IBM pruning.

Next, we consider combining techniques, specifically Stolcke pruning and a novel clustering technique. The clustering technique is surprising, in that it often first makes the model *larger* than the original word model. It then uses Stolcke pruning to prune the model to one that is *smaller* than a standard Stolcke-pruned word model of the same perplexity.

5.1 Previous Work

There are four well-known previous techniques for reducing the size of language models: count-cutoffs, weighted difference pruning, Stolcke pruning, and IBM clustering.

The best-known and most commonly used technique is count cut-offs. Recall from Eq. (4) that when creating a language model estimate for a probability of a word *z,* given the two preceding words *x* and *y*, a formula of the following form is typically used:

$$P(z \mid xy) = \begin{cases} \dfrac{C(xyz) - D(C(xyz))}{C(xy)} & \textit{if } C(xyz) > 0 \\ \alpha(xy) P(z \mid y) & \textit{otherwise} \end{cases}$$

In the count cut-off technique, a cut-off, say 3, is picked, and all counts *C(xyz)* $\le$ *3* are discarded. This can result in significantly smaller models, with a relatively small increase in perplexity.

In the weighted difference method, the difference between trigram and bigram, or bigram and unigram probabilities is considered. For instance, consider the probability *P(City|New York)* versus the probability *P(City|York)*, the two probabilities will almost be the same. Thus, there is very little to be lost by pruning *P(City|New York)*. On the other hand, in a corpus like *The Wall Street Journal, C(New York City)* will be very large, so the count would usually be pruned. The weighted difference method can therefore provide a significant advantage. In particular, the weighted difference method uses the value $[C(xyz) - D(C(xyz))] \times [\log P(z \mid xy) - \log P(z \mid y)]$. For simplicity, we give the trigram equation here; an analogous equation can be used for bigrams or other *n*-grams. Some pruning threshold is picked, and all trigrams and bigrams with a value less than this threshold are pruned. Seymore and Rosenfeld [1997] made an extensive comparison of this technique to count cut-offs, and showed that it could result in significantly smaller models than count cut-offs, at the same perplexity.

Stolcke pruning can be seen as a more mathematically rigorous variation on this technique. In particular, our goal in pruning is to make as small a model as possible, while keeping the model as unchanged as possible. The weighted difference method is a good approximation of this goal, but we can solve this problem exactly using a relative entropy-based pruning technique, Stolcke pruning. Stolcke [1998] showed that the increase in relative entropy from pruning is

$$-\sum_{x,y,z} P(xyz)[\log \bar{P}(z \mid xy) - P(z \mid xy)]$$

where $\bar{P}$ denotes the model after pruning, *P* denotes the model before pruning, and the summation is over all triples of words (*xyz*). Stolcke shows how to efficiently compute the contribution of any particular trigram *P(z|xy)* to the expected increase in entropy. A pruning threshold can be set, and all trigrams or bigrams that would increase the relative entropy less than this threshold are pruned away. Stolcke showed that this approach works slightly better than the weighted difference method, although in most cases, the two models end up selecting the same *n*-grams for pruning.

The last technique for compressing language models is clustering. In particular, Brown et al. [1990] showed that a clustered language model could significantly reduce the size of a language model with only a slight increase in perplexity. Let z^l represent the cluster of word *z*. The model is of the form $P(z^l|x^l y^l) \times P(z|z^l)$. To our knowledge, previous to our work, no comparison of clustering to any of the other three techniques has been done.

5.2 Pruning and Clustering Combined

Our new technique is essentially a generalization of IBM's clustering technique combined with Stolcke pruning. However, the actual clustering we use is somewhat different than might be expected. In particular, in many cases, the clustering we use first *increases* the size of the model. It is only after pruning that the model is smaller than a pruned word-based model of the same perplexity.

The clustering technique we use creates a binary branching tree with words at the leaves. By cutting the tree at a certain level, it is possible to achieve a wide variety of different numbers of clusters. For instance, if the tree is cut after the 8th level, there will be roughly 2^8=256 clusters. Since the tree is not balanced, the actual number of clusters may be somewhat smaller. We write z^l to represent the cluster of a word z using a tree cut at level l. Each word occurs in a single leaf, so this is a hard clustering system, meaning that each word belongs to only one cluster.

Consider the trigram probability $P(z|xy)$ where z is the word to be predicted, called the *predicted word*, and x and y are context words to predict z, called the *conditional words*. Either the predicted word or the conditional word can be clustered in building cluster-based trigram models. Hence there are three basic forms of cluster-based trigram models. When using clusters for the predicted word, as shown in Eq. (17), we get the first kind of cluster-based trigram model, called *predictive clustering*. When using clusters for the conditional word, as shown in Eq. (18), we get the second model, called *conditional clustering*. When using clusters for both the predicted word and the conditional word, we have Eq. (19), called *both clustering* (see Gao et al. [2001] for a detailed description).

$$P(z \mid xy) = P(z^l \mid xy) \times P(z \mid xyz^l) \tag{17}$$

$$P(z \mid xy) = P(z \mid x^j y^j) \tag{18}$$

$$P(z \mid xy) = P(z^l \mid x^j y^j) \times P(z \mid x^k y^k z^l) \tag{19}$$

We see that there is no need for the size of the clusters in different positions to be the same. We actually use two different clustering trees, one for the predicted position and one optimized for the conditional position [Yamamoto and Sagisaka 1999].

Optimizing such a large number of parameters is potentially overwhelming. In particular, consider a model of the type $P(z^l|x^jy^j) \times P(z|x^ky^kz^l)$. There are five different parameters that need to be simultaneously optimized for a model of this type: j, k, l, the pruning threshold for $P(z^l|x^jy^j)$, and the pruning threshold for $P(z|x^ky^kz^l)$. Rather than try a large number of combinations of all five parameters, we give an alternative technique that is significantly more efficient. Simple math shows that the perplexity of the overall model $P(z^l|x^jy^j) \times P(z|x^ky^kz^l)$ is equal to the perplexity of the cluster model $P(z^l|x^jy^j)$ times the perplexity of the word model $P(z|x^ky^kz^l)$. The size of the overall model is clearly the sum of the sizes of the two models. Thus, we try a large number of values of j, l, and a pruning threshold for $P(z^l|x^jy^j)$, computing sizes and perplexities of each, and a similarly large number of values of l, k, and a separate threshold for $P(z|x^ky^kz^l)$. We can then look at all compatible pairs of these models (those with the same value of l) and quickly compute the perplexity and size of the overall models. This allows us to relatively quickly search through what would otherwise be an overwhelmingly large search space.

Table I. Text Corpus Statistics

Text corpus	Training set (million characters)	Test set (million characters)
General-Newspaper	414	1
Magazines	292	1
Literature	10	1
Science-Tech-Newspaper	89	1
Filtered-Web-Data	31	0
IME	11	1
Computer-Press	3	1
Books	581	1
Raw-Web-Data	204	1
Open-Test	0	0.5
Total	1,640	8.5

6. RESULTS AND DISCUSSION

In this section we present the results of our main experiments; in Section 6.1 we describe the text corpus we used.

In Section 6.2 we show how lexicon and segmentation optimization works. We demonstrate the effectiveness of constructing a Chinese lexicon by automatically extracting words from a corpus. We then show that the iterative method of jointly optimizing a lexicon, segmentation, and language model not only results in better word segmentation over conventional approaches, but also improves the reduction in character perplexity of the language model.

In Section 6.3 we present experiments with optimizing training data. We show that our method of selecting training data yields better language models by using less training data. We then show that our method of adapting the training data domain outperforms simple, conventional language model adaptation approaches (e.g., combining data and combining models).

In Section 6.4 we give a fairly thorough comparison of different types of language model size reduction, including count cut-offs, weighted difference pruning, Stolcke pruning, and clustering. We then present results, using our novel clustering technique combined with Stolcke pruning, showing that it produces the smallest model at a given perplexity.

In Section 6.5 we present the overall system results in terms of the perplexity of the language model and character error rates (CER) in pinyin-to-character conversion. We show that the combination of methods, described in this article, yields the best results reported to date for Chinese SLM. We also present experiments that examine how perplexity is related to character error rate in pinyin-to-character conversion.

Table II. Statistics of the Open-Test Set

Open-Test	Data size (thousand characters)
Army	8.5
Computer	29.5
Culture	69.5
Economy	54.0
Entertainment	52.0
Literature	48.0
National	55.5
People	58.0
Politics	61.0
Science	30.0
Sport	57.0
Total	519.0

6.1 Corpus

The text corpus we used consists of approximately 1.6 billion Chinese characters, containing documents with different domains, styles, and times. The overall statistics of the text corpus are shown in Table I. Some corpora are fairly homogeneous in both style and domain, like Science-Tech-Newspaper, some are fairly homogeneous only in style but are heterogeneous in domain, like General-Newspaper and Literature; while still others are of great variety, like Magazines, Raw-Web-Data, Filtered-Web-Data, and Books. The IME corpus is balanced, collected from the Microsoft input method editor (IME, a software layer that converts keystrokes into Chinese characters). It consists of approximately 12 million characters that have been proofread and balanced among domains. There are two corpora collected from Chinese Websites: the Filtered-Web-Data corpus was verified manually and is of high quality and the Raw-Web-Data corpus is a large, mixed-quality set (it even has errors). The Raw-Web-Data corpus is used for experiments on training set optimization.

To evaluate our methods, from each corpus we built a disjoint test set of its corresponding training set, as shown in Table I. In addition, in most of our experiments we used a carefully designed and widely used independent Open-Test corpus As shown in Table II, it contains approximately half a million characters that have been proofread and balanced among domains, styles, and times.

Most of the character-error-rate results reported below were tested on this test set. We used a baseline lexicon with 50,180 entries, which was carefully defined by Chinese linguists.

6.2 Optimizing the Lexicon and Segmentation

In this section we first report the results of lexicon construction. We then show the performance of the iterative method for jointly optimizing the lexicon, segmentation, and language model. Combining these, we achieved better lexicons, better segmentation, and better language models. For future work, we show our preliminary studies on optimizing the feature form and parameter setting of the information gain-like metric for lexicon construction

Table III. Character Perplexity Results of a Bigram Using the Baseline Lexicon and the Extracted Lexicon

Lexicon	Size (KB)	PPc on Open-Test	PPc on training corpus
Baseline	53	62.68	33.64
Extracted	6	121.07	91.10
Extracted	10	84.25	54.22
Extracted	15	77.51	46.88
Extracted	20	74.09	42.93
Extracted	25	71.31	39.67
Extracted	30	70.06	38.31
Extracted	35	68.18	36.14
Extracted	40	66.55	34.26
Extracted	45	65.30	32.76
Extracted	50	64.56	31.69
Extracted	55	63.69	30.61

6.2.1 Lexicon Construction Results

In the first series of experiments, we compared the performance of the baseline lexicon and the lexicon extracted from the training corpus (consisting of 27 million characters, a mix of the General-Newspaper, Science-Tech-Newspaper, and Literature training sets) by the method described in Section 3. Our method's initial lexicon contained 6,763 frequently used Chinese characters. We used the training set itself and Open-Test as test sets. The character perplexities of the resulting corpus are shown in Table III: for bigram language models, we found that the extracted word/compound perplexity decreased as lexicon size increased from 6K to 55K. At the same size in the baseline lexicon, our method achieves similar performance. It turns out that a Chinese lexicon with comparable quality to a humanly compiled lexicon can be obtained automatically from a large training corpus using our method. Especially when using the training corpus as the test set, the extracted lexicon outperforms the baseline lexicon by reducing character perplexity by approximately 10%.

6.2.2 Joint Optimization Results

With a preliminary implementation of the joint optimization of lexicon, segmentation, and language model described in Section 3.2, we found that the system had improved our lexicon, and that numerous real words were missing in the humanly compiled lexicon. Some examples are shown in Table IV.

We also found that iterative improvement can correct many of the errors caused by the greedy maximum matching algorithm. For example, the maximum matching algorithm wrongly segmented "已开发和尚在开发的资源" into "已 \ 开发 \ 和尚 \ 在 \ 开发 \ 的 \ 资源" (the developed monk is developing resources), and after two iterations, our system produced the correct segmentation: "已 \ 开发 \ 和 \ 尚 \在 \ 开发 \ 的 \ 资源" (the developed and developing resource).

On average, we obtained about a 2-6% character perplexity reduction on the basis of this iterative refinement technique. We used all lexicons mentioned above as initial lexicons and tested the joint optimization method on the same training corpus. The initial language model at iteration 0 was bootstrapped by segmenting the sentences into words

Table IV. Examples of Newly Discovered Words

	Extracted words
Real words	粮库(grain depot), 编委(editorial committee), 作客(be guest), 自己(self), 跻身(ascend).
Debatable items	坐车(by bus), 驻足(make a temporary stay), 不法分子(badman), 秉公执法(execute the law justly).
Terms	光盘驱动器(CD-ROM Driver), 异步传输模式(asynchronous transfer model), 汪辜会谈(Wang-Gu talk).
Proper names	宣武门(XuanWu Gate), 爱丽舍宫(Elysee), 俄亥俄州(Ohio), 董建华(Dong Jianhua), 亚马逊(Amazon), 蔡元培（Cai Yuanpei）, 盖茨(Bill Gates).

Table V. Character Perplexity Results of a Bigram for 1-4 Iterations Using the Joint Optimization Method

Iteration	0	1	2	3	4
Character perplexity	38.31	37.56	37.32	37.31	37.31

using the lexicon–based maximum matching algorithm. Table V shows the training set character perplexity versus the number of iterations using the 30K lexicon as an example. It can be observed that the perplexity begins to saturate at the second iteration.

In addition, we believe our approach has the following benefits: (1) it gives a quantitative method for deriving the lexicon and segmentation, using perplexity as a consistent measure; (2) it minimizes error propagation from lexicon selection and segmentation; (3) it is extensible to any language where word segmentation is a problem.

6.2.3 More on Lexicon Construction: Feature Forms and Parameter Settings

In this section we examine the impact of different forms of interactive information. In addition, in order to extract an optimal lexicon of a given size, we try to find the optimal parameter setting (i.e., *MI*, *LSize*, *MaxL*, *RSize*, *MaxR*, and *RF*) of the information gain-like metric described in Section 3.1. We expect that, based on the optimal lexicon, the resulting language model has the lowest character perplexity. This is an ongoing research project at our lab. Some preliminary results were reported separately in Zhang et al. [2000] and Zhao et al. [2000].

The *mutual information* of two random variables *X* and *Y* is given by

$$MI(X,Y) = H(Y) - H(Y/X) = H(X) + H(Y) - H(X,Y) \tag{20}$$

where *H(.)* is the entropy. The mutual information between two symbols *x* and *y* is interpreted as

$$MI(x,y) = \log\frac{P(x,y)}{P(x)P(y)} \tag{21}$$

Table VI. Character Perplexity Results of Bigram Language Models Using a Baseline Lexicon and Lexicons Extracted with Various Equations.

Lexicon	Baseline	Lexicon extracted by different equations			
		(21)	(22)	(23)	(24)
Character perplexity on Test1	48.89	49.39	47.22	47.72	45.76
Character perplexity on Test2	100.32	101.53	98.61	99.07	98.05

Similarly, in our experiments, we also estimate the *Information Loss, IL(x,y)* of a *bigram (x, y),* using the following three forms:

$$IL(x,y) = \log \frac{P(x,y)}{P(x)+P(y)} \tag{22}$$

$$IL(x,y) = \log \frac{P(x,y)^{\alpha}}{P(x)P(y)} \tag{23}$$

$$IL(x,y) = P(x,y) \log \frac{P(x,y)}{P(x)P(y)} \tag{24}$$

where *P(.)* is the probability and α is the coefficient tuned to maximize the performance.

A series of experiments was conducted. The training corpus is a subset of the General-Newspaper training set, with approximately 50 million characters. The first test data (Test1) we used is another disjoint subset of the General-Newspaper corpus, with approximately 52 million characters. The second test data (Test2), containing 9 million characters, consists of documents from various domains, including shopping, news, entertainment, etc. (a mixture of the General-Newspaper, Magazines, and Books corpora). The results are presented in Table VI. We can see that by using Eq. (24) we achieved the best result, while by using Eq. (21), we obtained the worst result (even worse than the baseline lexicon). A rough explanation of the result is that, in case of a *bigram*, the information of the *bigram* relative frequency is very important in estimating the probability of the generation of a new word, and should act as a weighted factor of the relative entropy.

6.3 Optimizing Training Set Selection

In this section we evaluate two training set optimization methods, described in Section 4. Two corpora are used for experiments. The IME training set is used as the *seed set* (or in-domain corpus), which contains 11 million characters that were proofread and balanced among domains, and a mixture of the Filtered-Web-Data corpus and the Raw-Web-Data corpus, denoted WEB, is used as the *training set* (or out-of-domain corpus), which contains a total of 235 million characters collected from Chinese Websites.

We also discuss the problems of seed set selection and over-fitting.

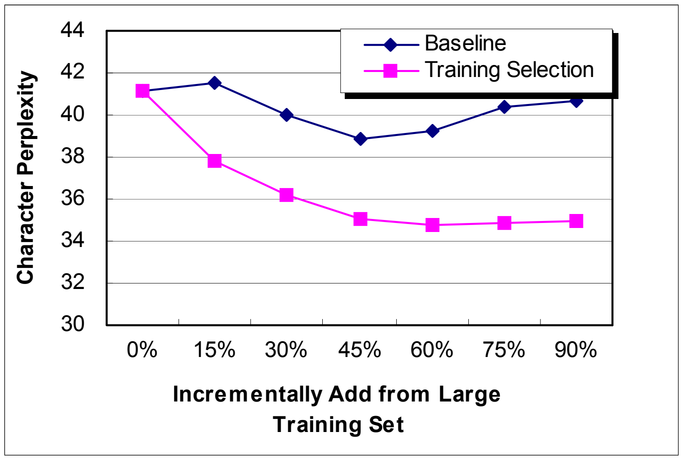

Fig. 6. Character perplexity results of trigram language models using the training set filtering method.

6.3.1 Training Set Filtering Results

In Figure 6 we display the performance of the training set filtering method. The perplexity results are obtained on the IME test set. Article boundaries for the seed set, training set, and test set are unknown. All resulting trigram language models were reduced to a fixed size of 35 megabytes. Baseline language models were built on the combination of seed set and a portion of training data randomly selected from the large training set. Using training set filtering, each time we incrementally added the best 15% of a training set for language model training. It turns out that the training set filtering method results in a series of language models with consistently lower character perplexities (up to a 12% reduction in character perplexity) than baseline models, given the same size of training data. Another interesting result is that, by using our method, the best language model was obtained when we used only approximately 60% of all the training data. We have some reasons for this. First, since the size of the language models is limited, as training data increases, it becomes saturated. Second, after the training set is ranked by quality, adding "bad" data (e.g., data with errors) could actually hurt performance.

We also used the Open-Test as the test set and repeated the experiments. The results are very similar, except that we obtained a much smaller perplexity reduction, i.e., up to 5%. This is due to the bigger difference between the seed set (i.e., IME corpus) and the test set (i.e., Open-Test). Additional experiments indicate that our method is more effective when the training set is a large, mixed-quality set (such as the one in this experiment).

6.3.2 Results of Adapting a Training Set

In this section we compare the performance of the training set adaptation method, called the *n*-gram distribution-based language model adaptation, described in Section 4.2, with other conventional domain adaptation methods.

Table VII. Perplexity and Character Error Rate Results for Open-Test Using Training Set Adaptation, Filtering, etc.

No.	Training set	Technique	Word perplexity	Character error rate	Character error rate reduction
1	IME	Baseline	645.24	7.05 %	
2	IME+WEB	BF	432.71	6.48 %	8.08 %
3	IME+WEB	SI	388.34	6.31 %	10.50 %
4	IME+WEB	OBF	411.77	6.21 %	11.91 %
5	IME+WEB	OSI	382.46	6.26 %	11.21 %
6	IME+WEB	DBA	389.87	5.80 %	17.73 %

In Table VII we show the perplexity and character error rate results, using Open-Test as the test set. The error rate results are obtained using the system of pinyin-to-character conversion described in Section 6.5. The baseline results were obtained using a trigram language model trained on the IME training set only. The second column contains the training data components; the third column shows the techniques we use; the fourth and fifth columns show the word perplexity and the character error rate of pinyin-to-character conversion, respectively; and the sixth column shows the percentage reduction in character error rate from the baseline.

Row 2 shows that simply adding out-of-domain training data, called the brute-force (BF) technique, results in 8.08% character error rate reduction. When we simply interpolate the IME trigram and the language model trained from out-of-domain training data, called the simple interpolation (SI) scheme, using Eq. (12), we obtain a slight improvement of 10.5% word error rate reduction. The most commonly used simple methods [Iyer et al. 1997; Clarkson and Robinson 1997; Seymore and Rosenfeld 1997] are based on the BF and SI techniques. We then combine BF and SI with the training data filtering method described in Section 4.1; that is, the out-of-domain training set has been filtered and does not contain data with errors. In Table VII we denote these two methods FBF and FSI, respectively. Further improvements are obtained, as shown in rows 4 and 5. Row 8 shows that the best results are obtained using the *n*-gram distribution-based language model adaptation (DBA) method, as described in Section 4.2. Note that the correlation between word perplexity and character error rate is not strong; while DBA has the lowest character error rate, OSI and SI have lower perplexities. This suggests that test set perplexity may not be a good criterion for estimating model interpolation weights.

6.3.3 Seed Set and Over-Fitting

For the training set optimization methods described in Section 4, the selection of the seed set is a common issue. In experiments, a seed set is usually defined as a set of high-quality corpora (e.g., available application documents or domain-specific documents). But, in practice, such a corpus is not large enough, and the resulting seed set is always unreliable. This can lead to the over-fitting problem, and in what follows we describe our approaches to it.

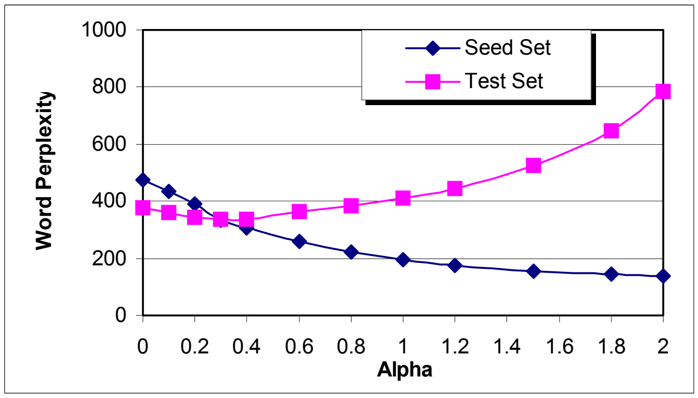

Fig. 7. Word perplexity results on a test set versus a seed set, using the training set adaptation method.

For filtering a training set, as described in Section 4.1.2, we iteratively increased the seed set by adding blind feedback until the resulting seed set was sufficient to train a robust language model.

For adapting a training set domain, we divided the seed set into partitions and used a cross-validation-like approach. For instance, in our experiments, we randomly divided the in-domain training data into five partitions, each of the same size. We picked one partition as a *test set* and combined the other four partitions as a *seed set*. Then we set α from 0.0 to 2.5, and combined bigram counts with Eqs. (14) to (16). Finally, a series of language models were estimated. As shown in Figure 7, as α increases, the perplexity on the seed set is reduced, and the perplexity on the test set is at first reduced and then rises sharply. Thus, we picked the value of α where the perplexity on the test set just begins to rise. We call the value at this point of α the *critical value* of α. We repeated the above experiments by defining different pairs of test sets and seed sets, and obtained the average critical value α.

6.4 Reducing Language Model Size

In this section we present a comparison of various techniques of language model compression, including count cut-offs [Jelinek 1990], weighted difference pruning [Seymore and Rosenfeld 1996], and Stolcke pruning [Stolcke 1998]. In previous work [Goodman and Gao 2000], we compared clustering without pruning [Brown et al. 1990] and showed that that technique is not as good as the others. In Gao et al. [2001], we also showed that conditional clustering models perform consistently worse than the baseline word-based trigram models, since the conditional clustering model always discards information for predicting words, and even with smoothing it does not bring any additional benefits. Hence we do not report the results of these two cases.

In this section we demonstrate our new technique, which combines a novel form of clustering with Stolcke pruning, yielding models that are typically 35% or more smaller than Stolcke pruning alone at the same perplexity. Most previous research on pruning

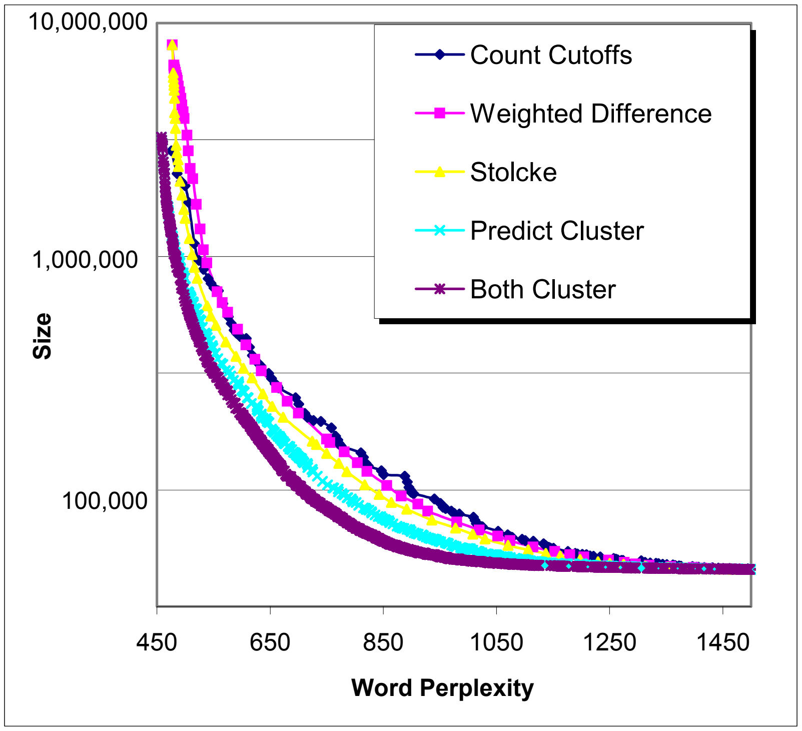

Fig. 8. Perplexity versus size on IME data..

language models was done on English. We performed all of our experiments on English, Chinese, and Japanese. Ours was the most comprehensive work done on English, and the most comprehensive, and perhaps the only, pruning experiments on Chinese and Japanese. For consistency, we report our results on Chinese only. English and Japanese results and a cross-language comparisons are reported by Goodman and Gao [2000] and Gao et al. [2001]. The results for English are qualitatively the same, although the improvement in our new techniques is quantitatively better for English, with typically a 50% reduction in model size versus Stolcke pruning for English alone, compared to a 35% reduction for Chinese.

For our experiments, described below, we built a large number of models. Rather than graph all points of all models together, we show only the outer envelope of the points. That is, if for a given model type and a given point there is some other point of the same type with both lower perplexity and smaller size than the first point, then we do not graph the first, worse point.

We built a very large number of models, as follows. We tried 167 different combinations for cut-offs, where the trigram cut-off varied between 0 and 1,024, in increments of about 50%, and the bigram cut-off was various fractions of the trigram cut-off, between 0.1 and 1.2 times the trigram cut-off. For weighted difference pruning and Stolcke pruning, we tried a large range of parameters, sufficient to cover the same perplexities as covered by the count cut-offs. For experiments with predictive clustering,

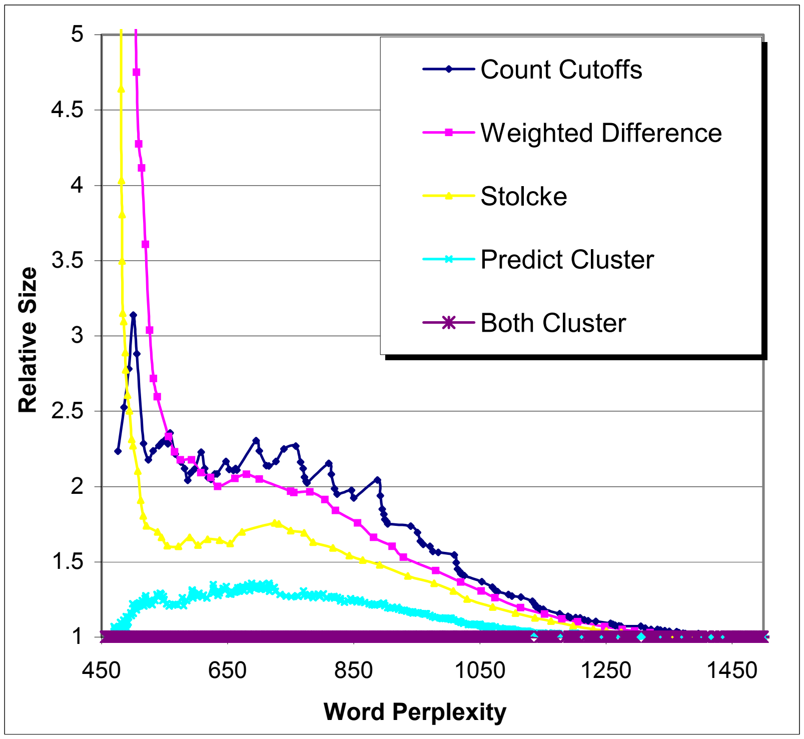

Fig. 9. Perplexity versus size on IME corpus.

we tried most models of the form $P(z^l|xy) \times P(z|xyz^l)$ for values $5 \leq l \leq 9$. For both clusterings, we tried most models of the form $P(z^l|x^jy^j) \times P(z|x^ky^kz^l)$ for values $6 \leq j \leq 15$, $8 \leq k \leq 16$, and $6 \leq l \leq 8$. The ranges were determined on the basis of pilot experiments. The size was measured as the total number of parameters of the system: one parameter for each bigram and trigram that was not a threshold, 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, approximately 50,000 words (some words in the vocabulary did not appear in the training data, and thus were discarded).

We performed experiments on the Chinese IME and Open-Test corpus, segmented by the baseline lexicon of 50,180 words. In particular, for training data, we used the IME training set. For heldout data, we used every 30th word, taken from the first 150,000 Figure 9 gives an alternate view of the same results. In order to show more detail, we plotted relative sizes. Each technique was plotted in comparison to both cluster techniques taken from a disjoint and nonoverlapping set of the Open-Test set.

Table VIII. Sample Parameter Settings for $P(z^l|x^jy^j) \times P(z|x^ky^kz^l)$

l	j	prune $p(z^l\|x^jy^j)$	k	prune $p(z\|x^ky^kz^l)$	word perplexity	size
7	7	8192	11	5120	1520	45578
8	10	768	10	384	964	50955
8	11	192	11	112	794	70746
8	14	256	11	64	760	80056
8	15	96	11	40	686	111370
8	13	48	13	42	643	150349
7	15	28	13	28	605	200889
7	13	16	13	16	558	302266
7	15	8	13	8	513	511891
7	all	6	13	4	493	753461
7	all	4	15	3	482	1010246
7	15	2	14	2	470	1523531
7	all	1.5	all	1.5	464	2064400
7	all	0.6	all	2	460	3005076
7	all	0.6	all	1.25	458	3249908

Figure 8 shows the results of these experiments. Unfortunately, due to the very large range of sizes, it was difficult to resolve details.

Figure 9 gives an alternate view of the same results. In order to show more detail, we plotted relative sizes. Each technique was plotted in comparison to both cluster techniques.

The main result here is that, over a wide range of values, both cluster techniques produce models that are at most 2/3 the size of Stolcke-pruned models with the same perplexity. At low perplexities, the models are half the size or less. This is a much greater improvement than Stolcke pruning over count cut-offs alone. The other interesting result is the surprisingly good performance of count cut-offs, better than previously reported, and, at low threshold values, marginally better than weighted difference pruning, and overlapping with Stolcke pruning. We have a few explanations for this.

First, Stolcke did not compare his technique directly to count cut-offs, but only to weighted difference pruning. A close look at the paper comparing weighted difference pruning [Seymore and Rosenfeld 1996] to count cut-offs shows that very few points were compared at the low-pruning end of the chart. Also, in the previous work, rather than trying a large number of reasonable values for bigram and trigram cut-offs, and then looking at the best ones, bigrams and trigrams were pruned in such a way as to have the same number of nonzero parameters.

The most interesting analysis is to look at some sample settings of the parameters of both cluster systems, as shown in Table VIII. The value "all" for *k* means that the tree was cut at infinite depth, i.e., each cluster contained a single word. The "prune" column indicates the Stolcke pruning parameter that was used.

First, notice that the two pruning parameters (in columns 3 and 5) tend to be very similar. This is good, since applying the theory of relative entropy pruning predicts that the two pruning parameters should actually have the same value.

Next, compare our model, of the form $P(z^l|x^jy^j) \times P(z|x^ky^kz^l)$, to traditional IBM clustering of the form $P(z^l|x^ly^l) \times P(z|z^l)$, which is equal to $P(z^l|x^ly^l) \times P(z|x^0y^0z^l)$. Traditional IBM clustering makes two assumptions that we see are suboptimal. First, it assumes that $j=l$. The best results come from unequal settings of j and l. Second, and more importantly, IBM clustering assumes that $k=0$. We see that not only is the optimal setting for k not 0, it is typically the exact opposite: *all* (in which case $P(z|x^k y^k z^l) = P(z|xyz^l))$, or 15, which is very similar. That is, we see that words depend on previous words, and that an independence assumption is a poor one. Of course, many of these word dependencies are pruned away, but, when a word does depend on something, the previous words are better predictors than the previous clusters. The other important finding here is that, for a great many of these settings, the unpruned model is actually larger than a normal trigram model: Whenever k=all, the unpruned model $P(z^l|x^j y^j) \times P(z|x^k y^k z^l)$ is actually larger than an unpruned model $P(z|xy)$.

This analysis of the data is very interesting. It implies that the gains from clustering are not from compression, but rather from capturing structure. Factoring the model into one in which the cluster is predicted first and then the word is predicted, given the cluster, allows the structure and regularities of the model to be found. This larger, better-structured model can be pruned more effectively.

6.5 System Results

We combined the above techniques and built a system for cross-domain general trigram SLM for Chinese. We trained the system with 1.6 billion characters of training data, as shown in Table I. In this section we present the perplexity results and the character error rates in pinyin-to-character conversion. We also investigate the correlation between perplexity and the error rate.

6.5.1 Perplexity Results

All bigram and trigram language models reported here are trained using techniques described in this article: that is, the optimized lexicon and segmentation and the Stolcke pruning methods. One thing worth mentioning here is the use of training set optimization. Since these language models are cross-domain general models, we could not define the so-called domain-specific seed set. So we did not use the training set adaptation method described in Section 4.2. We also note that there are some texts that are not suitable for language model training, such as low-quality data (e.g., data with errors) or ancient-style Chinese text. So we used the training set filter to throw out this "bad" data. We used the Total training set of 1.6 billion characters as seed set. As shown in Figure 10, we see that "bad" data stands out owing to perplexity ranking, since most training data is good, and the resulting seed language model is good enough.

In Table IX, we display the perplexity results of this system. Taking the trigram language model of size 200 megabytes as an example, we found that, across seven different domains, the total character perplexity was 33.19 (evaluated on a combination of all seven test sets). The lowest perplexity domain was General-Newspaper, with a perplexity of 24.68. The highest perplexity domain was Raw-Web-Data, with a perplexity of 44.32.

6.5.2 Conversion Results

We also evaluated the system for Chinese pinyin-to-character conversion. This is a similar problem to speech recognition, except that it does not take acoustic noise into account. The performance is generally measured in terms of character error rate, which is the number of character errors converted from Chinese pinyin divided by the number of

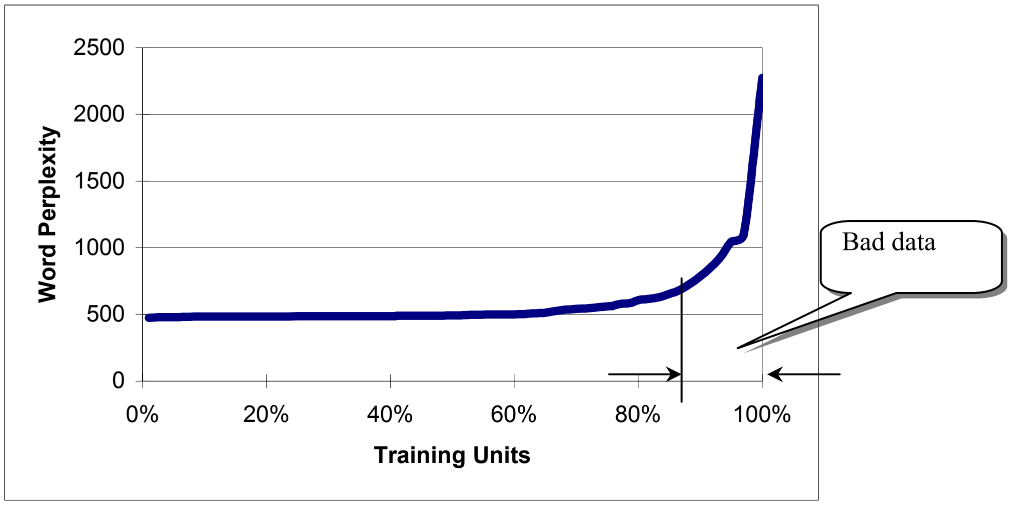

Fig. 10. Training units list ranked by perplexity.

Table IX. Combined System Character Perplexity Results

Test Set	Trigram				Bigram	
	200 MB	90 MB	30 MB	15 MB	30 MB	15 MB
Open-Test	32.76	36.98	41.62	45.54	56.56	57.86
IME	38.27	39.57	44.68	48.92	60.36	61.51
General-Newspaper	24.68	25.87	28.44	31.17	41.7	42.35
Magazines	25.54	27.47	30.49	33.4	56.63	44.35
Science-Tech Newspaper	29.12	30.85	34.05	37.25	47.91	48.7
Books	42.87	46.3	50.96	55.23	67.24	68.42
Raw-Web-Data	44.32	59.54	71.9	80.23	93.04	97.31
Total	33.19	36.63	41.19	45.16	56.63	57.88

characters in the correct transcript. The role of the language model is, for all possible word strings that match typed pinyin, to select the word string with the highest language model probability. Current products make about 10-20% errors in conversion of real data in a wide variety of domains. When using the Open-Test as the test set, our results are shown in Figure 11, along with MSPY2.0 as the baseline system, which is one of the best available products and delivers the best commercial accuracy today, in spite of minimal

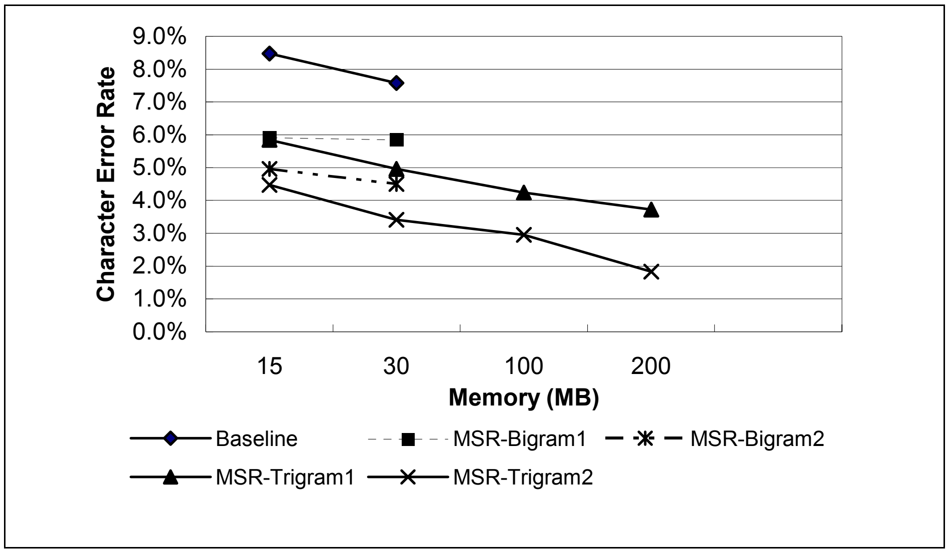

Fig. 11. Pinyin-to-text conversion result.

memory usage evaluated on the same data. In addition to the baseline, we show four different systems that we built. Table X summarizes the techniques we used for each language model system shown in Figure 11.

It turns out that compared to the commercial product, our system has up to a 50% lower error rate at the same memory size, and is about 76% better without any memory limits.

6.5.3 Perplexity, Entropy, and Chinese Pinyin-to-Character Conversion

In this section we briefly examine the performance of a language model measured in terms of perplexity/entropy with Chinese pinyin-to-character conversion using the language model. As mentioned earlier, pinyin-to-character conversion is similar to the speech recognition problem, and the performance is in terms of character error rates. Previous studies in speech recognition argued that there is some linear correlation between the error rate and entropy [Chen and Goodman 1999; Chen et al. 1998]. But it is well known that a low perplexity/entropy does not guarantee good performance in speech recognition because perplexity/ entropy do not take acoustic noise into account. So we often have to measure accuracy in speech recognition if the main concern is the contribution of the language model to acoustic pattern matching [Huang et al. 2000].

For these experiments, we collected the 11 trigram language models we built for the experiments mentioned above. Note that all these language models are not combined, since we have shown that there is no strong correlation between perplexity and character error rates for the combined language model in Section 6.3.2. They are trained on different corpora and are of different sizes. For each model, we first evaluated the character perplexity/entropy on the Open-Test set. We then calculated the character error rates on the Open-Test set for each of the 11 models in pinyin-to-character conversion, as described earlier.

Table X. Summary of Techniques In System Evaluation

	Baseline	MSR-Bigram1	MSR-Bigram2	MSR-Trigram1	MSR-Trigram2
Training Set	IME	Total	Total	Total	Total
Lexicon & Segmentation Optimization	NO	YES	YES	YES	YES
Training Set Filtering	NO	YES seed set: Total	YES seed set: Total	YES seed set: Total	YES seed set: Total
Training Set Domain Adaptation	NO	NO	YES seed set: IME training set	NO	YES seed set: IME training set
Pruning Method	Count Cutoff	Predict Cluster + Stolcke	Predict Cluster + Stolcke	Stolcke	Stolcke

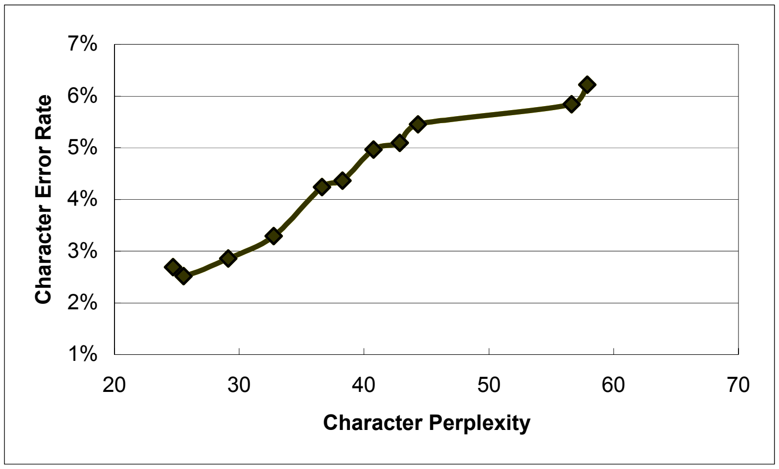

Fig. 12. Relation between perplexity and Chinese pinyin-to-character conversion character error rate on Open-Test set for 11 language models.

For consistency, in Figure 12, we only plotted the perplexity versus character error rate of the 11 models. We can see that, unlike the speech recognition system, the correlation between perplexity and character error rate is very strong. We also found that entropy correlates even better with the character error rate than with perplexity. This indicates that in some sense entropy may be a better measurement of language model performance.

It appears that perplexity/entropy and the character error rate correlate well when the corpus is changed, the training data size is changed, or the language model size is changed, but that more sophisticated changes, such as those in Section 6.3.2, do not necessarily lead to correlated changes.

7. CONCLUSION

In this article we presented a unified approach to Chinese statistical language modeling. This unified approach enhances *n*-gram-based Chinese statistical language modeling with automatic maximum-likelihood-based (perplexity-based) methods to select the lexicon, segment words, filter and adapt the training data, and reduce language model size.

We demonstrated improved results using each of these approaches. For lexicon and segmentation optimization, we showed that a Chinese lexicon that is comparable in quality to a manually generated lexicon can be obtained automatically from a large corpus on the basis of an information gain-like metric. We can achieve further improvement in terms of reducing character perplexity on the basis of the joint optimization of the lexicon, segmentation, and the language model. For optimizing training data, we demonstrated the effectiveness of the training set filtering method, which can select a suitable data set from raw Web data for training the language model. We also showed that an *n*-gram distribution-based method outperforms the conventional methods for adapting domains. For reducing language model size, we showed that by combining clustering and Stolcke pruning techniques in a novel way, we can achieve significantly better results than previous reduction techniques.

Finally, we combined all the techniques to build a system for statistical language modeling for Chinese. System evaluation demonstrated that the combined system produces the best-reported results (both in perplexity and character error rates in Chinese pinyin-to-character conversion) to date from a large, diverse corpus. We also showed that in pinyin-to-character conversion, there is a strong linear correlation between perplexity and character error rate, except for the case of the combined language model. This again demonstrates the effectiveness of the perplexity-based metric we used.

ACKNOWLEDGMENT

We thank Changning Huang, Hai-Feng Wang, Ming Zhou, Hsiao-Wuen Hon, Xuedong Huang, and other colleagues from Microsoft Research, and Lee-Feng Chien from Academia Sinica, Taiwan, for their help in developing the ideas and implementation in this article. We would also like to thank Dong-Feng Cai, Zheng Chen, Hao Li, and Hui Ding for their help in our experiments.

REFERENCES

BERTON, A., FETTER P., AND REGEL-BRIETZMANN, P. 1996. Compound words in large-vocabulary German speech recognition systems. *ICSLP96*.

BROWN, P. F., DELLA PIETRA V. J., DE SOUZA, P. V., LAI, J. C., AND MERCER, R. L. 1990. Class-based n-gram models of natural language. *Comput. Linguist.* 18, 467-479.

CHEN, S. F., AND GOODMAN, J. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech and Language* 13 (Oct.), 359-394.

CHEN, S. F., BEEFERMAN, D., AND ROSENFELD, R. 1998. Evaluation metrics for language models. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*.

CHIEN, L. F. 1997. PAT-tree-based keyword extraction for Chinese information retrieval. In *Proceedings of the ACM SIGIR'97 Conference* (Philadelphia, PA), 50-58.

CLARKSON, P. AND ROBINSON, A. 1997. Language model adaptation using mixtures and an exponentially decaying cache. In *Proceedings of the ICASSP-97 Conference*.

FUNG, P. 1998. Extracting key terms from Chinese and Japanese texts. *Int. J. Comput. Process. Oriental Lang. Special Issue on Information Retrieval on Oriental Languages*, 99-121.

GAO, J., LI, M., AND LEE, K. F. 2000a. N-gram distribution based language model adaptation. In *Proceedings of the ICSLP-2000 Conference* (Beijing, Oct. 16-20).

GAO, J., WANG, H. F., LI, M., AND LEE, K. F. 2000b. A unified approach to statistical language modeling for Chinese. In *Proceedings of the ICASSP-2000 Conference* (Istanbul, June).

GAO, J., GOODMAN, J., AND MIAO, J. 2001. The use of clustering techniques for language model application to Asian language. *Int. J. Comput. Linguist. Chinese Lang. Process*., 6, 1.

GIACHIN, E. P. 1995. Phrase bigrams for continuous speech recognition. In *Proceedings of the ICASSP-95 Conference*.

GOODMAN, J. AND GAO, J. 2000. Language model compression by predictive clustering. In *Proceedings of the ICSLP-2000 Conference* (Beijing, Oct.).

HEARST, M. 1997. TextTiling: Segmenting text into multi-paragraph subtopic passages. *Comput. Linguist.* 23, 33-64.

HUANG, X. D., ACERO, A., AND HON, H. 2000. *Spoken Language Processing*. Prentice Hall, Englewood Cliffs, NJ.

IYER, R., OSTENDORF, M., AND GISH, H. 1997. Using out-of-domain data to improve in-domain language models. *IEEE Signal Process. Lett.* 4, 8 (Aug.).

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, 450-506.

KATZ, S. M. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Trans. Acoustics, Speech Signal Process.* ASSP-35, 3 (March), 400-401.

LIN, S. C., TSAI, C. L., CHIEN, L. F., CHEN, K. J., AND LEE, L.S. 1997. Chinese language model adaptation based on document classification and multiple domain-specific language models. In *Proceedings of the 5th European Conference on Speech Communication and Technology* (Rhodes, Greece).

MANNING, C. D. AND SCHUTZE, H. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA.

MILLER, D., LEEK, T., AND SCHWARTZ, R. M. 1999. A hidden Markov model information retrieval system. In *Proceedings of the 22nd International Conference on Research and Development in Information Retrieval* (Berkeley, CA), 214-221.

ROCCHIO, J. J. 1971. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice Hall, Englewood Cliffs, NJ, 313-323.

SEYMORE, K., AND ROSENFELD, R. 1996. Scalable backoff language models. In *Proceedings of the International Conference on Speech and Language Processing*, Vol. 1 (Philadelphia, PA), 232-235.

SEYMORE, K., AND ROSENFELD, R. 1997. Using story topics for language model adaptation. In *Proceedings of the ICASSP-97 Conference*.

STOLCKE, A. 1998. Entropy-based pruning of backoff language models. In *Proceedings of the DARPA News Transcription and Understanding Workshop* (Lansdowne, VA.), 270-274.

TUNG, C. H., AND LEE, H. J. 1994. Identification of unknown words from a corpus. *Comput. Process. Chinese Oriental Lang.* 131-145.

WONG, P. K., AND CHAN, C. K. 1996. Chinese word segmentation based on maximum matching and word binding force. In *Proceedings of the 16th International Conference on Computational Linguistics* (Copenhagen), 200-203

WU, M. W. AND SU, K. Y. 1993. Corpus-based automatic compound extraction with mutual information and relative frequency count. In *Proceedings of the R.O.C. Computational Linguistics Conference* VI (Nantou, Taiwan), 207-216.

YAMAMOTO, H. AND SAGISAKA, Y. 1999. Multi-class composite n-gram based on connection direction. In *Proceedings of the ICASSP Conference* (Phoenix, AZ, May).

YANG, K. C., HO, T. H., CHIEN, L. F., AND LEE, L.S. 1998. Statistics-based segment pattern lexicon: A new direction for Chinese language modeling. In *Proceedings of the IEEE 1998 International Conference on Acoustic, Speech, Signal Processing* (Seattle, WA), 169-172.

ZHANG, J., GAO, J., AND ZHOU, M. 2000. Extraction of Chinese compound words: An experimental study on a very large corpus. In *Proceedings of the Second Chinese Language Processing Workshop* (Hong Kong, Oct. 8).

ZHAO, J., GAO, J., CHANG, E., AND LI, M. 2000. Lexicon optimization for Chinese language modeling. In *Proceedings of the ISCSLP-2000.* International Symposium on Spoken Language Processing *(*Beijing, Oct. 14-15).

ZUE, V. W. 1995. Navigating the information superhighway using spoken language interfaces. *IEEE Expert* 10, 5 (Oct.), 39-43.

Received December 2000; revised July 2001; accepted August 2001.