A Method to Build a Super Small but Practically Accurate Language Model for Handheld Devices

WU GenQing (吴根清) and ZHENG Fang (郑 方)*

Center of Speech Technology, State Key Laboratory of Intelligent Technology and Systems
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, P.R. China

E-mail: {wgq, fzheng}@sp.cs.tsinghua.edu.cn

Received September 26, 2002; revised April 29, 2003.

Abstract In this paper, an important question, whether a small language model can be practically accurate enough, is raised. Afterwards, the purpose of a language model, the problems that a language model faces, and the factors that affect the performance of a language model, are analyzed. Finally, a novel method for language model compression is proposed, which makes the large language model usable for applications in handheld devices, such as mobiles, smart phones, personal digital assistants (PDAs), and handheld personal computers (HPCs). In the proposed language model compression method, three aspects are included. First, the language model parameters are analyzed and a criterion based on the importance measure of n-grams is used to determine which n-grams should be kept and which removed. Second, a piecewise linear warping method is proposed to be used to compress the uni-gram count values in the full language model. And third, a rank-based quantization method is adopted to quantize the bi-gram probability values. Experiments show that by using this compression method the language model can be reduced dramatically to only about 1M bytes while the performance almost does not decrease. This provides good evidence that a language model compressed by means of a well-designed compression technique is practically accurate enough, and it makes the language model usable in handheld devices.

Keywords language model, language model compression, piecewise linear warping, rank-based quantization

1 Introduction

1.1 About Language Modeling

Language modeling is a commonly used means in information processing. It is often used in a decoding procedure from a given lower layer sequence $E = (e_1, e_2, \ldots)$ to an upper layer sequence $W = (w_1, w_2, \ldots)$, the goal of which is to find the maximum likely one among all possible upper-layer candidates, to provide a likelihood/probability measure for the upper layer. Suppose $M = (m_1, m_2, \ldots)$ (where $m_1 = 1$) is the index sequence for W, and $(e_{m_t}, e_{m_t+1}, \ldots, e_{m_{t+1}-1})$ is decoded into w_t $(t = 1, 2, \ldots)$. In such a decoding procedure, the same $(e_{m_t}, e_{m_t+1}, \ldots, e_{m_{t+1}-1})$ could be decoded into several different w_t's, and the same E could have several different index sequences M's, thus the decoding procedure can be regarded as *searching* a maximum likely sequence in an upper-layer sequence space, that is to say, $W_{ML} = \arg\max\limits_{W}\{P(W)|E \to W\}$ (where $\to$ means "*encoded into*"). Therefore, the decoding is also referred to as a *search* procedure. Several applications that use the language model are listed in Table 1. The use of language model is actually the utilization of context dependent information. A popular kind of language model is the statistical language model named the n-gram model[1], which is an $n-1$ order Markov process. In such a model framework, the current word is assumed to be dependent only on the preceding $n-1$ words. In other words,

$$P(w_t|w_1, \ldots, w_{t-1}) = P(w_t|w_{t-n+1}, \ldots, w_{t-1}) \tag{1}$$

In particular, the n-gram is called a *uni-gram*, *bi-gram*, or *tri-gram* model when $n = 1$, 2, or 3, respectively. For the bi-gram model, the probability of a word sequence $W = (w_1, w_2, L, w_T)$ is expressed as

$$P(W) = P(w_1)\prod_{t=2}^{T} P(w_t|w_{t-1}) \tag{2}$$

*The author currently is also with Beijing d-Ear Technologies Co., Ltd., fzeng@d-Ear.com.

Table 1. Applications Using Language Model

Layer / Application		Lower (E Seq.)	Middle (Lattice)	Upper (W Seq.)
LVCSR		Acoustic signal	Syllable/Phoneme	Word
Handwritten Char. Recognition (HCR)		Handwritten character	Character	Word
Optical Char. Recognition (OCR)		Printed character	Character	Word
Word Segmentation		Character	none	Word
	Pinyin	Letter	none	Word
Full Chinese Sentence IME	Digit	Digit	none	Word
	Stroke	Stroke	none	Word

Note: LVCSR stands for large vocabulary continuous speech recognition, IME for input method editor, and Pinyin is the pronunciation representation of the Chinese character.

while for the tri-gram model

$$P(W) = P(w_1)P(w_2|w_1)\prod_{t=3}^{T} P(w_t|w_{t-2}, w_{t-1}) \tag{3}$$

The bi-gram and tri-gram models are the most popular ones in LVCSR and some other applications. No matter in what kind of application, a language model contains *high-layer* context dependent information that is helpful to determine or choose the *best* result from all possible candidates. Because of this, many researchers are spending a lot of efforts in improving the performance of a language model in their corresponding application fields.

However, the use of language model encounters the following two difficulties.

The first one is the sparseness issue. For example, in Chinese there are about 50K frequently used words in LVCSR or Chinese full sentence IME applications. If we use the bi-gram language model, there will be $50,000^2 = 2.5 \times 10^9$ word pairs (or bi-grams for simplification); if we use tri-gram model, there will be $50,000^3 = 1.25 \times 10^{14}$ word pairs (or tri-grams). It is true that all such grams (bi-grams or tri-grams) do not occur equally and there will often be a large number of grams that cannot be seen or observed even in a huge training text corpus. This is called *sparseness*. However, the practical probability calculation requires that none of the gram probabilities be *zero*. How to assign a *reasonable* and relatively *accurate* probability value to an unseen gram is a big issue. It will directly affect the decoding accuracy.

The second is the problem of the big model size and the decoding efficiency resulted from it. It is well known that the bigger the training text corpus is, the more accurate the language model, and no doubt, the bigger the language model. Normally, for a 50K-word vocabulary, the size of a tri-gram model will be several hundred mega bytes. Such a big language model makes itself less usable. It costs a lot of static storage and runtime memory, occupies a lot of CPU resources, and lowers the search efficiency.

1.2 Popular Methods for Language Model Compression

The above two issues conflict with each other. A balance should be made between them. It seems easy to keep such a balance for applications in personal computer (PC) or better environments, and there are quite some researchers who are focusing on how to reduce the model size, or language model compression (LMC) in other words, with the accuracy of the language model kept, such as [2–5]. They have been achieving quite good results. These methods include count-cutoffs, pruning, and clustering.

The count-cutoffs method reduces the model size simply by throwing away those n-grams occurring less than k times, where k is a predefined constant[5,6].

The pruning method uses complex criteria to choose and throw away those less important n-grams. In the weighted difference method[7], the difference between the tri-gram and the bi-gram, or that between the bi-gram and the uni-gram, is taken as a kind of measurement for pruning. For example, if the probability values of $P(w_3|w_1, w_2)$ and $P(w_3|w_2)$ are almost the same, throwing away $P(w_3|w_1, w_2)$ will not cause too much distortion. In particular, this method uses the value

$$[(w_1, w_2, w_3) - D(C(w_1, w_2, w_3))] \cdot$$
$$[\log P(w_3|w_1, w_2) - \log P(w_3|w_2)]$$

to decide if it is necessary to keep the value $P(w_3|w_1, w_2)$ in the model. The Stolcke pruning[8] uses more mathematically rigorous criteria that use a kind of relative entropy-based technique. If the n-gram (w_1, w_2, w_3) is pruned, the increased entropy can be calculated as follows:

$$- P(w_1, w_2, w_3)* \\ [\log P'(w_3|w_1, w_2) - P(w_3|w_1, w_2)]$$

where P' denotes the model after pruning.

In spite of these compressing techniques, the compressed models are still a little bit too large to use practically.

Some researchers proposed word-class language model to reduce the model size and in the meantime to improve the model accuracy and robustness[3,8–10]. Taking bi-gram as an example, the idea can be expressed in the following equation:

$$P(W,C) = P(c_1)P(w_1|c_1)\prod_{t=2}^{T} P(c_t|c_{t-1})P(w_t|c_t) \tag{4}$$

where c_t means the word class (attribute) of word w_t, C is the word class sequence, $P(c)$ is the word class probability, $P(w|c)$ is the word-attribute probability and $P(c_t|c_{t-1})$ is the word-class transition probability. Because the number of word classes is very small, therefore the size of the model $\{P(c_i|c_j), P(w_k|c_i)|i, j, k\}$ is also very small. However, the difficulty in determining the number of word classes and the concrete word classes results in dissatisfying model accuracy.

The methods stated above work very well for PC-oriented applications, but for PDAs or HPCs, such a language model compressed via these techniques is even unusable because of the rigorous static storage, runtime memory, and computation ability limitations.

Naturally, a question arises: is it possible to reduce the size of a language model to less than 1 megabyte while the accuracy does not decrease dramatically? The answer as well as our solutions and results will be given in next sections.

2 Our Proposed Methods

2.1 Basic Idea

In this paper, we propose a novel LMC method from a different point of view.

Consider the question raised in the end of Section 1. The answer to this question mainly lies in the answer to the following question: is it possible to relatively accurately estimate the probabilities of unseen n-grams? If the answer is YES, the answer to the former question could also be YES.

An efficient way to re-estimate the probabilities of unseen n-grams is the Katz smoothing method, a kind of back-off algorithm[11], and its enhanced version[12]. The basic idea of this algorithm is to re-distribute or *discount* the occurrence probabilities of seen n-grams to those unseen n-grams according to their corresponding $(n-1)$-gram, recursively. Our previous proposal of big-discount re-estimation[6], where those n-grams with occurrence counts not greater than a predefined big constant are regarded as unseen ones, shows an improvement in language model performance. This reminds us that there is no inevitable connection between the size and the accuracy of a language model.

From this point on, we will present our LMC method with bi-gram as an example. In order to easily perform online language model adaptation[13] and to reduce storage amount, we store the occurrence counts (in integer number) instead of the occurrence probabilities (in float number) for uni-grams, and the probabilities can be calculated easily in real-time. The proposed method includes the following steps: (1) throwing away those bi-grams with less importance; (2) narrowing down the value range of the occurrence counts of uni-grams; and (3) assigning each bi-gram with a rank-based predefined bi-gram probability value. Obviously the performance of the compressed language model after all these steps is dependent on a well trained full language model, the seed language model.

2.2 Training Seed Language Model

The maximum likelihood estimation (MLE) is often used to estimate the parameters of the n-gram language model. Because a full language model with good performance is the footstone for LMC, the important thing to do before the LMC is to train a good language model from a huge text corpus.

After calculating the occurrence counts of all seen n-grams, we use the Katz back-off smoothing method to give the full language model parameters[11], and the bi-gram probability is given as

$$P_{Katz}(w_i|w_{i-1}) = \begin{cases} C(w_{i-1}, w_i)/C(w_{i-1}), & \text{if } r > r_T \\ d_r C(w_{i-1}, w_i)/C(w_{i-1}), & \text{if } 0 < r \leqslant r_T \\ a(w_{i-1})P_{Katz}(w_i), & \text{if } r = 0 \end{cases} \tag{5}$$

where $C(\cdot)$ stands for the occurrence count of the specified event, r stands for $C(w_{i-1}, w_i)$ for convenience, r_T is a count threshold for discounting purpose, and $a(w_{i-1})$ and d_r are the smoothing parameters for bi-gram. If n_r denotes the number of n-grams that occur exactly r times, d_r is calculated as follows:

$$d_r = \frac{\dfrac{r^*}{r} - \dfrac{(r_T+1)n_{r_T+1}}{n_1}}{1 - \dfrac{(r_T+1)n_{r_T+1}}{n_1}} \tag{6}$$

After d_r is determined, $a(w_{i-1})$ can be calculated as

$$a(w_{i-1}) = \frac{1 - \sum\limits_{w_i:r>0} P_{Katz}(w_i|w_{i-1})}{1 - \sum\limits_{w_i:r>0} P_{Katz}(w_i)} \tag{7}$$

The training process of the tri-gram model is similar to that of the bi-gram model. Because the seed language model contains almost all information extracted from the training corpus, the accuracy is of course quite high.

In order to reduce the model size to less than 1 mega bytes, the bi-gram model instead of the tri-gram model is used as the seed model in this paper.

2.3 Compressing the Seed Model

To get a high-performance compressed language model, we should preserve important information as much as possible and throw away less-important and redundant information. And practically, we should adopt some engineering methods to reduce the storage amount.

2.3.1 Choosing Important Bi-Grams

Generally speaking, a bi-gram model trained using a large corpus with several hundred million words contains tens of millions of bi-grams, actually each of which does not contribute equally. Based on this, not all of them are necessarily within an acceptable range of error. Those with less importance could be removed so as to diminish the model.

The very important thing in this step is the measure of importance. There are several alternatives for the measurement of importance to decide which to remove and which to preserve.

The first one is to use the joint probability to measure the importance. Given a bi-gram (w_1, w_2), the joint probability can be calculated as follows:

$$P(w_1, w_2) = \frac{C(w_1, w_2)}{\sum\limits_{w_1,w_2} C(w_1, w_2)} \tag{8}$$

The denominator of (8) remains the same for all bi-grams, so the importance of a bi-gram (w_1, w_2) can be measured by the occurrence count $C(w_1, w_2)$ equivalently. Assume $\{(w_1, w_2)|w_1, w_2\}$ is the whole bi-gram space. In this space, bi-gram (w_1, w_2) with a bigger occurrence probability $P(w_1, w_2)$ (or a bigger occurrence count $C(w_1, w_2)$) is referred to as the one with higher *importance*, and this measure is called an Importance Measure by Joint bi-gram Probability (IMJbP).

The second one is to use the following conditional probability to measure the importance of a bi-gram (w_1, w_2):

$$P(w_2|w_1) = \frac{C(w_1, w_2)}{C(w_1)} \tag{9}$$

That is to say, the bi-gram with a bigger conditional probability is referred to as one with higher *importance*. We call this measure an Importance Measure by Conditional bi-gram Probability (IMCbP). Roughly thinking, the IMCbP should be more reasonable than the IMJbP. It can be seen from the following example. Consider two bi-grams, where $C(w_{11}, w_{12}) = 1$ with $C(w_{11}) = 10$ and $C(w_{21}, w_{22}) = 1$ with $C(w_{21}) = 1{,}000$. The counts of these two bi-grams are equal, therefore their joint probabilities are equal, but the counts of their corresponding histories are quite different. Because $C(w_{11})$ is much less than $C(w_{21})$, bi-gram $C(w_{11}, w_{12})$ should be much more important than bi-gram $C(w_{21}, w_{22})$ relatively.

Based on the above analysis, IMCbP is used as the importance measure. Only those bi-grams with conditional probability greater than a predefined threshold will be kept in the compressed model.

On the other hand, we should also check whether some of the preserved bi-grams are redundant. Let us see an example. Suppose the back-off smoothing algorithm[11] gives the kept bi-gram (w_1, w_2) a re-estimated probability approximately the same as its initial probability estimated from the training corpus, that is to say, either way leads to almost the same result, or one of these two is redundant. In this situation, it is unnecessary to keep this bi-gram because its probability can be calculated using the back-off algorithm.

By this kind of checking, some redundant bi-grams can be removed.

After these two steps in the first-stage processing, the model size will become much smaller while the performance of the model is almost preserved.

2.3.2 Compressing Uni-Gram Count Value Range

It is necessary to use multiple bytes to present the occurrence count of a uni-gram in the seed language model trained using a huge training corpus. Usually, a 4-byte word (long integer) is used for uni-gram. This will obviously increase the model size because we store the counts instead of the probabilities in our language model. An alternative way to reduce the model size is to store the count value with short integer, which suggests us to compress the count values into a small range.

The famous Harvard linguistic professor George Kingsley Zipf issued the classical law about the statistical characteristic of language. It shows that the frequency of occurrence of some event (p), as a function of the rank (i) when the rank is determined by the above frequency of occurrence, is a power-law function $p_i \gg 1/i^a$ with the exponent a close to [14], as shown in Figs.1–3. From this law, it can be concluded that most n-grams occur with very low frequencies, and our previous experiments have also proven this fact[14].

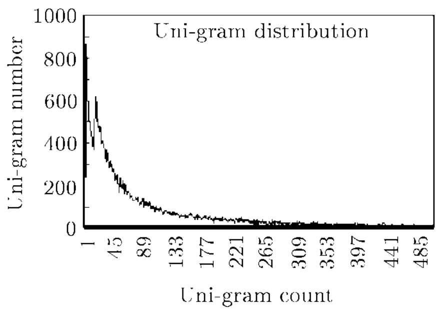

Fig.1. Uni-gram distribution.

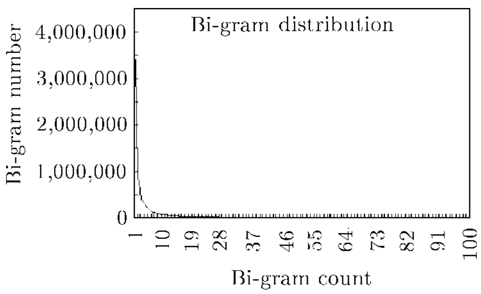

Fig.2. Bi-gram distribution.

That is to say, though a 4-byte integer is selected to store the occurrence count of a uni-gram, most of the uni-grams have very small occurrence counts. This is very similar to the voice signal in the telephone line where the amplitudes of most of the sample data are very small. In the voice signal processing, the A-law or the μ-Law is adopted to compress the signal. The basic idea of the A-Law or the μ-Law is to compress a linear PCM sample (13 bits) down to 8 bits (one byte). Such an idea can be borrowed into the compressing of the n-gram counts, in other words, to compress the count value according to a warping curve. In order to simplify the calculation, we use a piecewise linear function as follows:

$$C' = \begin{cases} C, & C \leqslant C_0 \\ C_0 + s \cdot (C - C_0), & C > C_0 \end{cases} \tag{10}$$

where C_0 is a connecting point that all values less than C_0 will remain unchanged, and s is the slope that is always much less than 1, as illustrated in Fig.4. This is referred to as a *piecewise linear warping method.*

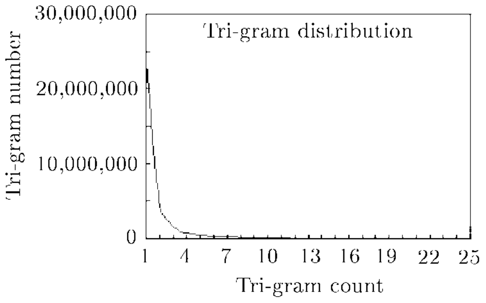

Fig.3. Tri-gram distribution.

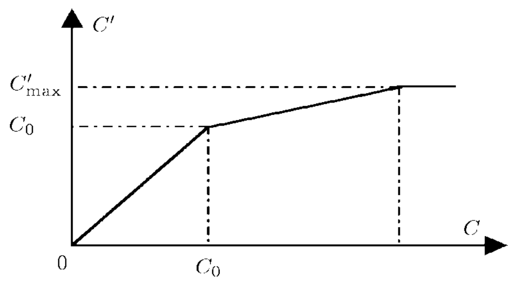

Fig.4. Piecewise linear compression curve.

2.3.3 Approximating Bi-Gram Probabilities by Rank

Above, we use a piecewise linear compression function for uni-gram count compression. Anyway this method is not suitable for bi-gram count compression because compressing the bi-gram count value range will result in a much bigger accuracy loss. To avoid this, we choose to approximate the bi-gram probabilities directly.

The main idea here is to approximate the occurrence probability according to its rank instead of its actual value so that no probability values are stored because the rank-related probabilities can be trained offline and they are fixed instead of dynamic. An easy way to estimate the probabilities is to use a codebook. The calculation is simple as follows. Suppose there are n bi-grams $(g|w_1)$ sharing the same history w_1 in the compressed model. We call such a w_1 an *n-style history*. Sort these bi-grams in a descending order of the bi-gram count values, and the number i-th bi-gram will be assigned with a probability $P_{i,n}^{\text{code}}$, which can be calculated offline as

$$P_{i,n}^{\text{code}} = \frac{\sum\limits_{w_1:N_{w_1}=n} P_i(\cdot|w_1)}{\sum\limits_{w_1:N_{w_1}=n} 1} \tag{11}$$

where $P_i(\cdot|w_1)$ is the probability of the number i-th bi-gram with the history w_1 in the full language model and N_{w_1} the number of bi-grams with history w_1 in the compressed model. The assigned value is actually the average probability of the i-th n-gram with n-style history. This method is very effective because it considers not only the position of the n-gram but also the number of the n-grams with the same history. We call the proposed method a *rank-based quantization method*.

The statistical results of the codebook are listed in Table 2.

Table 2. Probability Codebook of Bi-Grams

n \ i	0	1	2	3	4
1	0.385				
2	0.322	0.157	–	–	–
3	0.285	0.139	0.091		
4	0.273	0.132	0.084	0.060	–
5	...	...	...	...	...

2.3.4 *Real-Time Probability Accessing*

After all the above three steps, a compressed language model is now available. The piecewise linear warping method is used to compress the uni-gram counts and the rank-based quantization method is used to approximate the bi-gram probabilities. Therefore the probability of the compressed model could be accessed as follows. If the target bi-gram can be found in the compressed language model, the probability will be accessible in the lookup table, i.e., the probability codebook; otherwise, the back-off algorithm will be performed over the compressed uni-gram counts to give the probability.

As mentioned in Section 1, the decoding procedure can be regarded as *searching* a maximum likely sequence in an upper-layer sequence space, and it will lead to tremendously frequent accessing to the model. In order to meet the real-time requirements in a PDA, the main part of the model is stored in the disk or the flash card, and a hash cache mechanism, which stores the most latest frequently used unit in the memory cache pool, is used. This technique improves the accessing speed at least 5 times and reduces the memory cost to about 20K bytes.

As a matter of fact, the compressed language model is not normalized. In other words, the sum of the probabilities of all the n-grams with the same history does not equal 1.0, actually, it is only a little bit greater than or less than 1.0. The probability normalization is an easy thing to do, however our primary experiments show that the normalization does not improve the performance of the compressed model. This is also explainable theoretically. Practically, what affects the model performance more is the relative relation between any two bi-grams with the same history instead of the absolute value of each bi-gram.

2.3.5 *Influence of Vocabulary Size*

It is well known that in Chinese a sentence is a sequence of words *without* any separate denotation between the adjacent words. Chinese characters are the basic units for Chinese language, and a Chinese word may contain one character or more. This fact will lead to a serious situation that the vocabulary is very flexible. First, the set of Chinese characters can be used as the vocabulary directly. Because the common character set size is not too large (usually about 6,700), the language model size is relatively small. Actually, in real applications, some characters always occur simultaneously in a specific order and represent a particular concept, and they are used to form a multi-character word, thus the vocabulary will be extended to contain multi-character words. Obviously, the vocabulary size is determined mainly by the number of the multi-character words.

The model is trained in such a process. First, the corpus is segmented into word streams according to the vocabulary. Second, each word is assigned with an identity number and the model is trained. It can be seen that the language model size will increase with the vocabulary size; similarly, the performance of the model will increase with the increase of the vocabulary too. Considering this situation, suppose w_1 and w_2 are two words in a small vocabulary and the bi-gram (w_1, w_2) is not in it but in a large vocabulary. There is often such an example. Bi-gram (w_2, w) appears in an LM with a smaller vocabulary while bi-gram $(\overline{w_1w_2}, w)$ appears in an LM with a larger vocabulary. From the smaller vocabulary point of view, $(\overline{w_1w_2}, w)$ is taken as a tri-gram (w_1, w_2, w). This example shows a bi-gram in an LM with a large vocabulary may contain the same information as a tri-gram in LM with a small vocabulary. In other words, the context-dependent information can be embedded in words instead of word n-grams when using a large vocabulary, which will obviously reduce the size of the language model when the amount of information is the same.

Therefore, increasing the vocabulary size can also act as an alternative method to reduce the model size. In our system, the vocabulary size is about 50K.

3 Experiments

The full tri-gram language model, i.e., the seed language model, used in this paper is trained with a large corpus containing about 200 million Chinese words. The corpus includes 4 years' text data from *the People's Daily* (of the years of 1993, 1994, 1996, and 1997) and some texts from other newspapers. The vocabulary is made up of 50,624 Chinese words, where there are 6,201 monosyllable words (12.3%), 37,976 bi-syllable words (75.0%), 1,615 tri-syllable words (3.2%) and 4,832 quad-syllable words (9.5%)[15].

After the full language model is compressed, the resulted language model has the size of less than 1 mega bytes.

Three test corpora are chosen. *Corpus A* (35,025 characters) is a political lecture given by the Chinese President JIANG Zemin. *Corpus B* (1,800 sentences with 23,310 characters) is taken from the *Chinese National High-Tech R&D 863 Project*, and *Corpus C* (375 sentences with 3,466 characters) includes news taken from the web site of *PhoenixTV in Hong Kong* (http://www.phoenixtv.com.cn).

The application using which the comparison experiments are done is a full Chinese sentence Pinyin-to-character conversion system, i.e., a full Chinese sentence IME via Pinyin.

Experimental results are given in the following.

3.1 Character Error Rate Against Model Size

Experiments are done to show how the model performance changes while the model is compressed from a quite large one to a very small one. Fig.5 gives the curve of the character error rate against model size. It shows that the model performance keeps almost unchanged when the size is compressed to less than 10M bytes. As it can be seen from the figure, the error rate of *Corpus A* is very low, so *Corpus A* must be very similar to the training corpus, and it can be called as a low-perplexity corpus, while *Corpus C* is a high-perplexity one. The experiment result indicates that the compression method works very well for a high-perplexity corpus (such as *Corpus C*), because the polyline for *Corpus C* in Fig.3 is very flat.

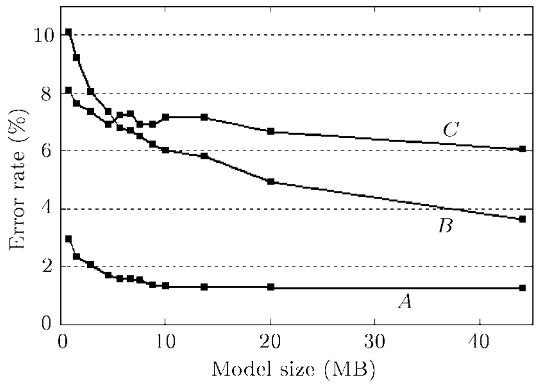

Fig.5. Character error rate against model size.

3.2 Performance of Uni-Gram Count Compression

In this experiment, only the uni-gram model is used during the decoding procedure. The seed language model uses 4-byte integer (Long Word) to store the uni-gram occurrence counts. For the compressed language model, we compare the use of 2-byte integer (Word) and 1-byte integer (Byte) to store the uni-gram counts in the piecewise linear compression.

As shown in Table 3, using Word or Byte to store the uni-gram counts in the piecewise linear compression method leads to an absolute accuracy decrease of 0.3% or 1.8%, respectively, compared with the full language model. The accuracy decrease can be ignored practically.

Table 3. The Conversion Accuracy (%) Using the Piecewise Linear Compression Function for Uni-Gram Counts

Corpus \ Method	A	B	C	Average
Seed model	92.02	83.51	86.79	87.44
Word width ($C'_{\max} = 65,535$)	91.89	82.95	86.50	87.11
Byte width ($C'_{\max} = 255$)	91.11	81.55	84.39	85.68

3.3 Performance of Compressed Model

Shown in Table 4 is the performance comparison among the compressed model, the full tri-gram model and the full bi-gram model. Comparing the compressed bi-gram model with the full tri-gram model, we can see that the model size decreases to 0.3% and the average accuracy decreases by only 4.1%.

Table 4. The Conversion Accuracy (%) of the Compressed Language Model Compared with the Full Language Model

Corpus \ Model	A	B	C	Average
Full tri-gram model (Size: 340MB)	99.34	98.90	94.23	97.49
Full bi-gram model (Size: 43MB)	98.75	96.39	93.94	96.36
Compressed bi-gram (Size: 940KB)	97.07	90.94	92.03	93.35

4 Analysis and Conclusions

In this paper, after analyzing the purpose and factors of a language model, we propose a new language model compression method with three techniques: an importance measure used to determine which n-grams to keep and which to remove, a piecewise linear warping method used to compress the uni-gram count value range, and a rank-based quantization method to re-estimate the bi-gram probability values.

By using the novel language model compression method, we can compress the model from several hundred megabytes to less than 1 megabyte while the performance is almost not reduced.

Why the result is so satisfying? The explanation could be as follows.

As a matter of fact, a language model is used to describe the co-occurrence probabilities of any words. Therefore what affects its performance is the description ability of words' co-occurrence probabilities no matter how large the model is. Based on this, a good solution to reducing the language model size is to preserve the high-density information contained in a language model. The reason why the proposed method achieves such a good result lies in that it throws away the unusable, less-usable and redundant information as much as possible while only the most usable information is kept.

The proposal of such a language model compression method makes the language model usable for applications in handheld devices, such as mobiles, PDAs, and handheld PCs, where storage and memory are luxuries.

References

[1] Jelinek F, Mercer R L. Interpolated estimation of Markov source parameters from sparse data. In *Pattern Recognition in Practice*, Gelsema E S, Kanal L N (eds.), Amsterdam, North-Holland, 1986.

[2] Di S, Zhang L, Chen Z *et al.* N-gram language model compression using scalar quantization and incremental coding. In *International Symp. Chinese Spoken Language Processing* (*ISCSLP'2000*), Beijing, China, 2000, pp.347–350.

[3] Goodman J. Language model size reduction by pruning and clustering. In *Int. Conf. Spoken Language Processing* (*ICSLP'2000*), Beijing, China, 2000.

[4] Whittaker E, Raj B. Quantization-based language model compression. *EuroSpeech*, Aalborg, Denmark, 2001, pp.33–36.

[5] Jelinek F. Self organized language modeling for speech recognition. In Readings in Speech Recognition, Waibel A, Lee K F (eds.), Morgan Kaufmann, 1990.

[6] Zheng F, Wu J, Song Z J. Improving the syllable-synchronous network search algorithm for word decoding in continuous Chinese speech recognition. *J. Computer Science & Technology*, Sept. 2000, 15(5): 461–471.

[7] Seymore K, Rosenfeld R. Scalable backoff language models. In *Int. Conf. Spoken Language Processing* (*ICSLP'1996*), Vol.1, Philadelphia, 1996, pp.232–235.

[8] Stolcke A. Entropy-based pruning of backoff language models. In *Proc. DARPA News Transcription and Understanding Workshop*, Lansdowne, VA, 1998, pp.270–274.

[9] Yan P J, Zheng F, Xu M X *et al.* Word-class stochastic model in a spoken language dialogue system. In *Int. Symp. Chinese Spoken Language Processing* (*ISCSLP'2000*), Beijing, Oct. 13–15, 2000, pp.141–144.

[10] Niesler T R, Woodland P C. Variable-length category-based n-grams for language modeling. Technical Report, Cambridge University, UK, April 1995.

[11] Katz S M. Estimation of probabilities from sparse data for the language model component of a speech recognizer. In *Int. Conf. Acoustics, Speech and Signal Processing* (*ICASSP'1987*), 1987, 35(3): 400–401.

[12] Wu G Q, Zheng F, Wu W H *et al.* Improved Katz smoothing for language modeling in speech recognition. In *Int. Conf. Spoken Language Processing* (*ICSLP'2002*), Vol.2, Denver, 2002, pp.925–928.

[13] Wu G Q, Zheng F, Jin L, Wu W H. An online incremental language model adaptation method. *EuroSpeech*, Aalborg, Denmark, Sept. 3–7, 2001, 3: 2139–2142.

[14] Zipf G K. Selective studies and the principle of relative frequency in language. Harvard University Press, Cambridge, MA, 1932.

[15] Zheng F. A syllable-synchronous network search algorithm for word decoding in Chinese speech recognition. In *IEEE Int. Conf. Acoustics, Speech and Signal Processing* (*ICASSP'1999*), Phoenix, USA, March 15–19, 1999, pp.II-601–604.

WU GenQing is currently a Ph.D. candidate of Center of Speech Technology, the State Laboratory of Intelligent Technology and Systems, Tsinghua University. He received his B.S. degree in computer science and technology from the Department of Computer Science and Technology, Tsinghua University, in 1999. He is now focusing on language modeling for speech recognition. His current research interests include language modeling, language model adaptation and language model compression techniques.

ZHENG Fang is currently an associate professor of Tsinghua University. He is the Director of Center of Speech Technology, State Laboratory of Intelligent Technology and Systems. Dr. Zheng graduated from the Department of Computer Science and Technology of Tsinghua University and received his B.S., M.S. and Ph.D. degrees from Tsinghua University, in 1990, 1992 and 1997 respectively. He has been working in speech recognition and understanding at the Department of Computer Science and Technology, Tsinghua University, since 1988, and now is with the State Key Laboratory of Intelligent Technology and Systems. He has published over 110 technical papers on acoustic/language modeling, isolated/continuous speech recognition, keyword spotting, dictating, language understanding, and so on. He is now an IEEE member, an ISCA member, a member of the Artificial Intelligence and Pattern Recognition Technical Commission of China Computer Federation, a member of the Editorial Committee of the *Journal of Chinese Information Processing*, and a key member of Oriental-COCOSDA. He is serving as a reviewer of several domestic and international journals. Recently, he has been the General Chair of Oriental-COCOSDA'2003, a member of the Scientific Committee of ISCA Tutorial and Research Workshop (ITR-Workshop) on Pronunciation Modeling and Lexicon Adaptation for Spoken Language Technology 2002, and a member of the Technical Committee and International Advisory Committee of the Joint International Conference of the Fifth Symposium on Natural Language Processing (CNLP) and '2002 Oriental COCOSDA Workshop (SNLP-O-COCOSDA).