International Symposium on Chinese Spoken Language Processing 2002, pp. 339-342, Aug. 22-24, 2002, Taipei

A COMPRESSION METHOD USED IN LANGUAGE MODELING FOR HANDHELD DEVICES

Genqing WU, Fang ZHENG[1]*, and Wenhu WU*

Center of Speech Technology, State Key Laboratory of Intelligent Technology and Systems
Department of Computer Science and Technology, Tsinghua University, Beijing, 100084, China
[wgq, fzheng, wuwh]@sp.cs.tsinghua.edu.cn, http://sp.cs.tsinghua.edu.cn

ABSTRACT

In this paper, a new n-gram language model compression method is proposed for applications in handheld devices, such as mobiles, PDAs, and handheld PCs. Compared with the traditional methods, the use of the proposed method can compress the model to a great extent with good performance preserved. The proposed method includes three aspects. The language model parameters are detailedly analyzed and a criterion based on the probability and the importance of n-grams is used to determine which n-grams should be kept and which be removed. A curving compressing function is proposed to be used to compress the n-gram count values in the full language model. And a code table is extracted and used to estimate the probabilities of bi-grams. Our 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 makes the language model usable in handheld devices.

1. INTRODUCTION

The statistical language model (LM), which is one of important parts in automatic speech recognition, is usually trained from a very large text corpus using the Markov model [1], where the word sequence is considered as an observation of an *n*-1 order Markov process called *n*-gram. In such a model framework, the current word is dependent only on the preceding *n*-1 words. For example, for tri-gram where *n*=3, given a word sequence $s = w_1 w_2 ... w_n$, the probability can be estimated as $P(s) = P(w_1) P(w_2|w_1) \prod_{i=3}^{n} P(w_i|w_{i-2} w_{i-1})$. For this kind of n-gram language model, such methods as the maximum likelihood estimation (MLE) are often used to estimate the model parameters.

Nowadays, language modeling researchers mainly focus on two aspects, to improve the modeling accuracy and to reduce the model size. Ways to improve the accuracy include enhancing the smoothing, estimating the probabilities of the unseen n-grams more precisely, and combining the traditional n-gram framework with other methods such as probabilistic latent semantic analysis (LSA). All these methods work but none of them is remarkable enough. Furthermore, according to the recent literature, the performance of the language model increases greatly with the increase of the size of the training corpus. On the other hand, with the booming development of handheld devices, such as mobiles and PDAs and handheld PCs, it is a great trend that more and more PC based applications will be ported to handheld devices. Obviously, the memory and storage limitations of handheld devices will be the bottleneck of the porting and those applications using the huge language model are encountering unprecedented difficulties. In this situation, some researchers focus on the compression of the language models, such as [2] and [3], and achieve quite good performance. However, due to the storage and calculation limitations of the handheld devices, they are still a little bit too large to use practically.

In this paper, we conduct an elementary research on language model compression method for handheld devices, which is different from the traditional techniques. This paper is organized as follows. Firstly, a full language model for use in PC environments is trained by an elaborate process; secondly, three techniques are used to compress the language model to a much smaller one, and finally, experiments are done and the results are given and analyzed.

2. TRAINING OF FULL LANGUAGE MODEL

The basic idea of language model compression is to keep as much information as possible in the compressed model with a much smaller size. The performance of a compression method is often measured by the performance degeneration during the compression. So a full language model with good performance is the footstone for language model compression.

In our research, we use the Katz smoothing method to train the full language model [4]. As we know, the traditional Katz smoothing method smoothes the bi-gram model this way

$$P_{Katz}(w_i \mid w_{i-1}) = \begin{cases} C(w_{i-1}, w_i) / C(w_{i-1}), & if \quad r > r_T \\ d_r C(w_{i-1}, w_i) / C(w_{i-1}), & if \quad 0 < r \le r_T \\ \alpha(w_{i-1}) P_{Katz}(w_i), & if \quad r = 0 \end{cases} \quad (1)$$

where $C(\cdot)$ stands for the occurring 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, $\alpha(w_{i-1})$ 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

$$d_r = \frac{\frac{r^*}{r} - \frac{(r_T+1)n_{r_T+1}}{n_1}}{1 - \frac{(r_T+1)n_{r_T+1}}{n_1}} \quad (2)$$

[1] The author currently is also with Beijing d-Ear Technologies Co., Ltd., *fzheng@d-Ear.com*.

After d_r is determined, $\alpha(w_{i-1})$ can be calculated by this formula

$$\alpha(w_{i-1}) = \frac{1 - \sum_{w_i : r > 0} P_{katz}(w_i \mid w_{i-1})}{1 - \sum_{w_i : r > 0} P_{katz}(w_i)} \quad (3)$$

The training process of tri-gram is similar to that of bi-gram. Because the full language model contains almost all information extracted from the training corpus, the accuracy is of course quite high. And the language model compression is based on such an accurate full language model aiming at reducing the model size with performance kept.

3. COMPRESSION METHOD

As mentioned above, the main idea of language model compression is to keep as much information as possible when compressing the original full model. In such a compressing method, the important information should be kept while the redundant information should be taken away as much as possible.

According to such a requirement, the language model size should be about 1M or even less so that it could be used in handheld devices. It's obvious that the tri-gram language model is not suitable and nor applicable in such situation. Accordingly, we propose to use bi-gram model with size reduced greatly enough for practical use.

3.1 Choosing Important Grams

Generally speaking, a bi-gram model trained from a large corpus with several hundreds million words contains tens of millions of bi-grams, actually each of which does not contribute equally. Based on this, not all of them is necessary within an acceptable range of error, some could be selectively removed so as to minish the model. There are several methods for the decision on which to remove and which to preserve.

The first method measures the importance of a bi-gram according to the joint probability. Given a bi-gram (w_1, w_2), the joint probability can be calculated as follows

$$P(w_1, w_2) = \frac{C(w_1, w_2)}{\sum_{w_1, w_2 \in W} C(w_1, w_2)} \quad (4)$$

where W is the vocabulary. We can find easily that the denominator of equation (4) remains same for any bi-gram, and therefore the count of bi-gram (w_1, w_2) is equivalent to the joint probability. If we suppose w_1 and w_2 be independent and define $\{(w_1, w_2) \mid w_1 \in W, w_2 \in W\}$ as the whole event space, obviously, $P(w_1, w_2)$ is the occurring probability of the event (w_1, w_2), and so we refer to an event with a bigger probability as one with higher *importance*.

The second method measures the importance of a bi-gram according to the conditional probability, and it can be calculated from the following equation

$$P(w_2 \mid w_1) = \frac{C(w_1, w_2)}{C(w_1)} \quad (5)$$

Intuitively, the second method is more reasonable than the first one. It can be illustrated as follows. Considering two simple situations, where in situation 1, $C(w_{11}, w_{12})$=1 and $C(w_{11})$=10 while in situation 2, $C(w_{21}, w_{22})$ = 1 and $C(w_{21})$ = 1000. The counts of grams (w_{11}, w_{12}) and (w_{21}, w_{22}) are equal, therefore their joint probabilities are equal, but the counts of their corresponding histories are quite different. Because $C(w_{11})$ in situation 1 is much less than $C(w_{21})$ in situation 2, relatively, the gram (w_{11}, w_{12}) for w_{11} is much more important than gram (w_{21}, w_{22}) for w_{21}.

The two methods above consider only the counts or the probabilities of the bi-grams; intuitively, if we consider bi-grams and uni-grams together, the choosing should be more reasonable. For example, suppose bi-gram (w_1, w_2) is very important to the model and will be kept in the compressed model according to the above criterion, suppose the back-off smoothing calculation gives a similar probability estimation, that is to say, either way leads to an almost same result, or one of these two is redundant. In this situation, it is unnecessary to keep the bi-gram because the probability can be calculated using the back-off method.

After this first-stage processing, the model size will be smaller while the performance is preserved.

3.2 Compressing Count Values

Because the training corpus is quite large, for the full language model, it is necessary to use multiple bytes to present the occurring count of an n-gram, and usually, a 4-byte double word (long integer) is used for uni-gram. This will obviously increase the model size. An alternative way to reduce the model size is to store the count value with short integer, which suggests us to compress the count value range.

The famous Harvard linguistic professor George Kingsley Zipf issued the classical law about the statistical characteristic of language [5], and it shows that 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 $p_i \approx 1/i^a$ with the exponent *a* close to unity [5]. From this law, it can be concluded that most of the n-grams occur with very low frequencies, and our experiments provide proofs for it.

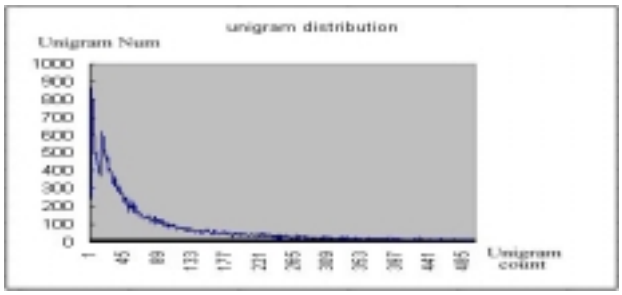

Figure 1: Uni-gram Distribution

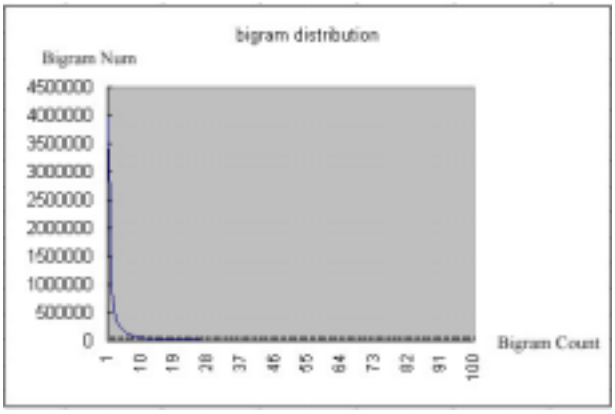

Figure 2: Bi-gram Distribution

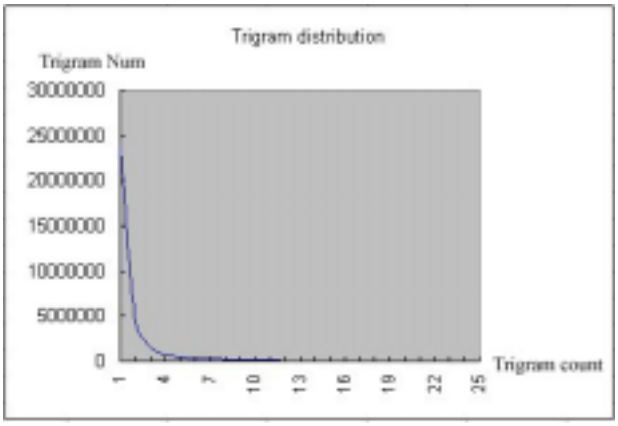

Figure 3: Tri-gram Distribution

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

$$C' = \begin{cases} C & C \le C_0 \\ C_0 + s \bullet (C - C_0) & C > C_0 \end{cases} \tag{6}$$

where C_0 is a connecting point, all values less than C_0 will be unchanged. S is the slope and is always much less than 1, as illustrated in Figure 4. In Section 5, the experimental results for the piecewise linear compression method used for uni-gram count compression will be given.

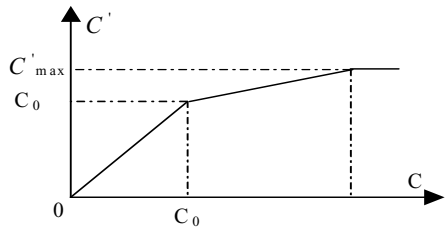

Figure 4: Piecewise Linear Compression of Count Value

3.3 Approximating Probability according to Rank

The piecewise linear compression function is usually used for uni-gram count compression, and a powerful method can be used for bi-gram compression.

In the full language model, the bi-gram (w_1, w_2) is corresponding to the occurring count and Equation (1) is used to calculate the probability. However, given the size of the compressed model, more bi-grams can be stored if the count information is discarded. The main idea here is to approximate the occurring probability according to its rank instead of its actual value. In this case, no probability value is needed to be 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 as simple as follows. Suppose there are n grams sharing the same history w_1 in the compressed model. These grams will be sorted in a descending order of bi-gram counts, and the number i bi-gram will be assigned with a probability $P_{i,n}^{code}$, which can be calculated offline as

$$P_{i,n}^{code} = \frac{\sum_{w_1 : N_{w_1} = n} P_i(\bullet \mid w_1)}{\sum_{w_1 : N_{w_1} = n} 1} \tag{7}$$

where $P_i(\bullet \mid w_1)$ 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. Table 1 gives the statistical result of the codebook.

Table 1: Probability Codebook of Bi-grams

n \ i	0	1	2	3	4	5
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	0.262	0.130	0.083	0.058	0.045	…
6	…	…	…	…	…	…

4. PRACTICAL CONSIDERATIONS

Summarily speaking, for the compressed model, the probabilities will be calculated in this way: if the bi-gram can be found in the full model, the probability will be assigned with the corresponding value in the codebook; otherwise, the compressed uni-gram model will give the probability.

In our application, piecewise linear compression method is used to compress the uni-gram counts and a codebook is used to approximate the bi-gram probabilities, both of the two methods destruct the normalization of the model, which means that the summation of the probabilities of the n-grams with a same history does not equal 1 (actually, it is only a little bit greater or less than 1). It is easy to normalize it, but our primary experiments show that the normalization doest not improve the compressed model. This is also explainable theoretically. What affects the model performance more is the relative relation between any two grams instead of the absolute value of each gram.

5. EXPERIMENTS AND RESULTS

The full tri-gram language model used in our experiments is trained from a large corpus containing about 200 million Chinese words. The corpus covers 4-year text data of *People's Daily* (from 1993 to 1994 and from 1996 to 1997) and a few texts of other newspapers. The vocabulary consists of 50,622 Chinese words. The full model is then compressed into a small model of about 1 MB size.

Three test corpora are predefined. 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 from the *Chinese National High-Tech 863Project*, and Corpus C (375 sentences with 3,466 characters) is news from the web of PhoenixTV in Hong Kong (http://www.phoenixtv.com.cn).

The compressed model is tested across a Chinese Pinyin-to-character (note: Pinyin is the pronunciation of a character conversion system, and the results are given in the following sections.

5.1 Performance of the Curving Method

Table 2: The Accuracy (%) of Conversion Using Piecewise Linear Compression Method for Uni-gram Count

Method \ Corpus	A	B	C	Avr.
Uncompressed model	92.02	83.51	86.79	87.44
WORD Curving	91.89	82.95	86.50	87.11
BYTE Curving	91.11	81.55	84.39	85.68

In this experiment, only uni-gram is used for decoding. The uncompressed model use 4-byte data to store the occurring counts of uni-grams. As shown in Table 2, using WORD (2 bytes) or BYTE and the piecewise linear compression method can achieve almost the same accuracy as uncompressed uni-gram model.

5.2 Compressed Model Performance

Table 3: The Conversion Accuracy (%) of Compressed Model Compared with Uncompressed Model

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

Table 3 shows the performance of compressed model compared with the full tri-gram model and the bi-gram model. It can be seen that the compressed model is much smaller than the full tri-gram model and the bi-gram model, but the performance is comparable and good enough for applications in handheld devices.

6. CONCLUSIONS

With the booming development of the handheld devices, more and more applications should and will be ported to handheld devices. In this paper, a method for language model compression is proposed. Because of the poor storage and computation ability of the devices, the compression ratio of the method should be very high, to achieve that, the model is shrunken from the full model by a selective method, which analyzes the parameters of the model carefully and uses both the conditional probability and the importance of the unit to determine whether it should be reversed or not. Additionally, a curving technique is designed to reduce the size of the uni-gram counts. Furthermore, a codebook is used to restore the bi-gram probabilities. Our experiments show that though the size of the compressed model is only about 1 MB, it is good enough to be used in handheld devices.

7. REFERENCES

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

[2] Shuo DI, Lei ZHANG, Zheng CHEN, Eric CHANG, Kai-Fu LEE, "N-Gram Language Model Compression Using Scalar Quantization and Incremental Coding", *International Symposium on Chinese Spoken Language Processing*, Beijing, China, 2000, pp.347-350.

[3] Whittaker E, Raj B, "Quantization-based Language Model Compression", *Proceedings of Eurospeech Conference*, Aalborg, Denmark, 2001, pp. 33-36.

[4] S.M. Katz, "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer," *ICASSP'87,* 35(3): pp. 400~401, 1987.

[5] G. Zipf, Selective Studies and the Principle of Relative Frequency in Language. Harvard University Press, Cambridge, MA, 1932.