Journal of Shanghai University (English Edition), 2003, 7(2): 148~155
Article ID: 1007-6417(2003)02-0148-08

A Bit Progress on Word-Based Language Model

CHEN Yong(陈勇), *CHEN Guo-Ping*(陈国评)
Department of Computer Science and Information System, University of Hong Kong, China

Abstract A good language model is essential to a postprocessing algorithm for recognition systems. In the past, researchers have presented various language models, such as character based language models, word based language model, syntactical rules language model, hybrid models, *etc*. The word N-gram model is by far an effective and efficient model, but one has to address the problem of data sparseness in establishing the model. Katz and Kneser *et al*. respectively presented effective remedies to solve this challenging problem. In this study, we proposed an improvement to their methods by incorporating Chinese language-specific information or Chinese word class information into the system.

Key words language model, pattern recognition, Chinese character recognition.

1 Introduction

Language model (LM) is commonly used in post processing algorithm of recognition system to improve the recognition rate[1,2]. During postprocessing, language model serves as a knowledge base to assist the recognition system to make a better decision. To achieve this goal, the language model should contain linguistic information about a language. According to their content, language models can be categorized into two classes, syntactical language model and statistical language model. Syntactical language model usually contains some syntactical rules that the language should follow. On the other hand, the statistical language model is usually formulated as a probability distribution to reflect how likely a string of words occur as a sentence. Usually the statistical model is implemented as an N-gram model based on the Markov model theory (also called (N-1)th order Markov model). In an N-gram language model, conditional probabilities of different N-gram patterns are collected over a training data. In practice, due to the limited memory resource, the value of N is usually less than or equal to 3. If we focus on characters in the language (In Chinese language, the word is made up of one or more Chinese characters), we call the N-gram language model as character N-gram model. If we focus on words of the language, we call the N-gram language model as a word N-gram model. If we focus on word classes of the language, we call the language model as word class N-gram language model. Word N-gram language model is significantly performs better than a character N-gram language model and word class language model, since the word N-gram language model more accurately reflecs the intrinsic statistical characteristics of the language than the other two. However, to establish the word N-gram language model, one has to solve the problem of data sparseness in the training process. It is common that many word N-gram patterns would not appear in a huge amount of training data, even though the size of vocabulary is medium and the value of N is equal to 2. Those unseen N-gram patterns will cause zero-valued conditional probabilities in the language model which will cause problem during application of the language model. To solve this problem, a smoothing technique is needed. Generally, there are two kinds of smoothing techniques, interpolating and backing-off. In interpolating smoothing techniques, the N-gram language model is combined with lower order model, say (N-1)th-gram, (N-2)th-gram, *etc*. Usually lower order models have probabilities greater than zero when N-gram model is zero. The combined result will then be greater than zero. In backing-off smoothing techniques, two techniques are involved. First, a probability mass will be formed by reserving and conglomerating some probability pieces from overestimated probabilities. Second, distribute the probability mass over unseen patterns. To distributing the proba-

Received Aug. 6, 2002; Revised Feb. 19, 2003
CHEN Yong, Ph. D. Candidate, Email: yongchen@csis.hku.hk

bility mass reasonably, we have to back off to the lower order model. Both Katz smoothing and Kneser-Ney smoothing discussed in this study belong to this kind of backing-off technique.

Katz solved data sparseness by backing off to (N-1) th-gram model and distributing probability mass according (N-1)-gram probability distribution upon occurrence of an unseen N-gram pattern. Kneser and Ney distributed the probability mass according to the number of (N-1)-gram patterns which can stand before the rear word of N-gram pattern to form a seen N-gram patterns. After studying Katz's and Kneser-Ney's methods, we think that we can improve their methods by incorporating Chinese language-specific linguistic information or word class bigram information. The language-specific linguistic information is one kind of special character bigram information. We use two kinds of information to guide the distribution process of probability mass. The guided distribution effect is better than the original methods.

This paper is structured as follows. In the next section, we introduce some background knowledge involved in establishing the statistical language model. In the third section, we introduce smoothing techniques, including some existing ones and our improved versions of Katz smoothing and Kneser-Ney smoothing. In the fourth section, we present experimental results and the discussion. The fifth section is the conclusion of this paper.

2 N-gram Language Model Basics

The N-gram language model is usually formulated as a probability distribution $p(s)$ over strings s that attempts to reflect how likely a string s occurs as a sentence. For example, there is a sentence s composed of the words $w_1 \cdots w_l$, we can express $p(s)$ as

$$\begin{aligned} p(s) &= p(w_1)p(w_2 \mid w_1) \cdot \\ &\quad p(w_3 \mid w_1 w_2) \cdots p(w_l \mid w_1 \cdots w_{l-1}) \\ &= \prod_{i=1}^{l} p(w_i \mid w_1 \cdots w_{i-1}) \end{aligned}$$

For bigram models ($N=2$), assuming a first order Markov model, we can make the approximation that the probability of a word depends only on the identity of the immediately preceding word, giving us

$$p(s) = \prod_{i=1}^{l} p(w_i \mid w_1 \cdots w_{i-1}) \approx \prod_{i=1}^{l} p(w_i \mid w_{i-1}) \quad (1)$$

To estimate conditional probability $p(w_i \mid w_{i-1})$, we can simply count how often the bigram $w_{i-1}w_i$ occurs in some text and normalize. Let $c(w_{i-1}w_i)$ be the number of times the bigram $w_{i-1}w_i$ occurs in the given text. Then, we can have

$$p(w_i \mid w_{i-1}) = \frac{c(w_{i-1}w_i)}{\sum_{w_i} c(w_{i-1}w_i)} \quad (2)$$

The text available for building a model is called training data. For bigram models, training data typically consists of millions of words. In our experiment, the amount of training data is 10 000 000 Chinese characters. The value for $p(w_i \mid w_{i-1})$ given in Eq. (2) is called the maximum likelihood (ML) estimation.

We can generalize Eq. (1) to Eq. (3) for cases of $N>2$,

$$p(s) = \prod_{i=1}^{l+1} p(w_i \mid w_{i-n+1}^{i-1}) \quad (3)$$

where w_i^j denotes the words $w_i \cdots w_j$ and where we take w_{-n+2} through w_0 to be <Bos> and w_{l+1} to be <Eos>. To estimate the probabilities $p(w_i \mid w_{i-n+1}^{i-1})$, the analogous equation to Eq. (2) is

$$p(w_i \mid w_{i-n-1}^{i-1}) = \frac{c(w_{i-l+1}^{i})}{\sum_{w_i} c(w_{i-n+1}^{i})} \quad (4)$$

The most widely used statistical language model is N-gram language model. In this paper we will focus on word bigram model($N=2$).

3 Smoothing

It is not unusual that a particular bigram pattern did not appear in a training data. As a result, the conditional probability of it would be zero. However, zero-valued bigram pattern would incur problem in applications. For example, in character recognition postprocessing, one attempts to find a path l in a character lattice that maximize the posteriori probability $p(l \mid s) = \frac{p(s \mid l)p(l)}{p(s)}$. If $p(l)$ is zero, then $p(l \mid s)$ will be zero and the path l will never be considered as a recognition result, regardless of how unambiguous the recognizing result is. Thus, whenever a path l causing $p(l)=0$ occurs during a character recognition, an error may occur. Assigning all strings a nonzero probability helps prevent errors in an application.

Smoothing techniques are used to address this prob-

lem. Furthermore, smoothing techniques can help us get more accurate statistical characteristics of a language. There exist many smoothing techniques. In this paper, we introduce the Good-Turing's method, the Katz's method and the Kneser-Ney's method. The Katz's method is based on the Good-Turing's method.

3.1 Good-Turing's method

In Good-Turing's method[3], the key insight suggested by Good and Turing is the introduction of the notion, frequency of frequency, *i.e.* the number of bigram which occur r times. It can be denoted by the notation N_r.

Good and Turing used the frequency of frequency in Eq. (5) to calculate a corrected count of a bigram pattern, which is usually a bit smaller than the original value.

$$r^* = (r+1)\frac{N_{(r+1)}}{N_r} \tag{5}$$

where r^* is the corrected count of the bigram pattern, while r denotes the original count. When a bigram pattern is unseen, then $r=0$. According to Eq. (5), the corrected count will be greater than zero since N_0 and N_1 both are greater than 0. Hence, with this additional information, frequency of frequency, Good and Turing preclude zero value count.

Please note the result r^* of Eq. (5) is not the conditional probability of the bigram pattern before further processed by Eq. (2). r^* is the numerator on the right hand side of Eq. (2).

3.2 Katz smoothing

Katz's smoothing technique includes two aspects, producing probability mass by discounting conditional probabilities of overestimated bigram patterns and distributing probability mass over conditional probabilities of underestimated bigram patterns. This smoothing technique can be formulated as Eq. (6).

$$p_{Katz}(w_{i-1}w_i) = \begin{cases} r/C(w_{i-1}), & \text{if } r>k \\ d_r r(w_{i-1}w_i)/C(w_{i-1}), & \text{if } 0<r\leqslant k \\ \alpha(w_{i-1})p_{ML}(w_i), & \text{if } r=0 \end{cases} \tag{6}$$

where

(1) r denotes the original count of any bigram pattern in a training data,

(2) $\alpha(w_{i-1})$ is a normalization factor that makes the total number of counts in the $\sum_{w_i} C_{Katz}(w_{i-1}w_i)$ unchanged, *i.e.*, $\sum_{w_i} C_{Katz}(w_{i-1}w_i) = \sum_{w_i} C(w_{i-1}w_i)$. Katz defined $\alpha(w_{i-1})$ in Eq. (7).

$$\alpha(w_{i-1}) = \frac{1-\sum_{w_i:C(w_{i-1}w_i)>0} p_{Katz}(w_i|w_{i-1})}{1-\sum_{w_i:C(w_{i-1}w_i)>0} p_{ML}(w_i)} \tag{7}$$

(3) d_r denotes the discounting factor, which can be obtained by Eq. (8),

$$d_r = \frac{\frac{r^*}{r} - \frac{(k+1)n_{(k+1)}}{n_1}}{1-\frac{(k+1)n_{(k+1)}}{n_1}} \tag{8}$$

where

- r^* is the corrected count derived by Eq. (5).
- k is a threshold, usually k is 5.
- $n_{(k+1)}$ is the frequency of frequency $(k+1)$.
- n_1 is the frequency of frequency 1, *i.e.*, the number of patterns that occur once.

(4) $d_r r$ denotes the corrected count.

According to Eq. (6), if a bigram pattern is seen in the training data and its count greater than k, then its count will not be discounted. If a bigram pattern is seen in the training data less than or equal to k times, then its count will be discounted by the factor d_r.

Now, let us take a close look at Eq. (7). First let us focus on the numerator. Suppose that there is a vocabulary $V = \{w_1, w_2, \cdots, w_n\}$ including n words. For a word $w_a \in V, 1\leqslant a\leqslant n$, we assume that there are four bigram patterns with w_a as the front word found in the training data. They are $w_a w_i$, $w_a w_j$, $w_a w_k$ and, $w_a w_l$, w_i, w_j, w_k, $w_l \in V$; $i\neq j\neq k\neq l\neq a$. w_i, w_j, w_k and w_l form a set V^s, *i.e.*, $V^s = \{w_i, w_j, w_k, w_l\}$. And, $V^u = V - V^s$. V^u contains words, which are never seen following the word w_a in the training data. Let p_i^{∇}, p_j^{∇}, p_k^{∇} and p_l^{∇} respectively be contributes from the original conditional probabilities p_i, p_j, p_k and p_l of $w_a w_i$, $w_a w_j$, $w_a w_k$ and $w_a w_l$ to the probability mass. Hence, we have, $p_i^{\nabla} = (1-d_{r_i}) \times p_i$, $p_j^{\nabla} = (1-d_{r_j}) \times p_j$, $p_k^{\nabla} = (1-d_{r_k}) \times p_k$ and $p_l^{\nabla} = (1-d_{r_l}) \times p_l$. According to the numerator of Eq. (7), we have,

$$\begin{aligned} &1-[p_{Katz}(w_i|w_a) + p_{Katz}(w_j|w_a) + \\ &\quad p_{Katz}(w_k|w_a) + p_{Katz}(w_l|w_a)] \\ &= 1-[(p_i - p_i^{\nabla}) + (p_j - p_j^{\nabla}) + \end{aligned}$$

$$(p_k - p_k^{\nabla}) + (p_l - p_l^{\nabla})]$$
$$= 1 - [(p_i + p_j + p_k + p_l) -$$
$$(p_i^{\nabla} + p_j^{\nabla} + p_k^{\nabla} + p_l^{\nabla})]$$
$$= p_i^{\nabla} + p_j^{\nabla} + p_k^{\nabla} + p_l^{\nabla}$$
$$= probability\ mass$$

the dominator of Eq. (7) can be expanded as

$$1 - \sum_{w_i: C(w_{i-1}w_i)>0} p_{ML}(w_{i-1}w_i)$$
$$= 1 - [p_{ML}(w_i) + p_{ML}(w_j) + p_{ML}(w_k) + p_{ML}(w_l)]$$
$$= \sum_{w_u: w_u \in V^u} p_{ML}(w_u)$$

As illustrated in the above example, the numerator of Eq. (7) corresponds to the probability mass while the dominator of Eq. (7) corresponds to the sum of ML estimates of probabilities of all words in the set V^u. Thus, when $r = 0$, according to Eq. (6), we have,

$$\alpha(w_{i-1})p_{ML}(w_i) = \frac{probability\ mass}{sum\ of\ ML\ estimates\ of\ prob.\ of\ words\ in\ V^u} \cdot p_{ML}(w_i) = probability\ mass \cdot \frac{p_{ML}(w_i)}{sum\ of\ ML\ estimates\ of\ prob.\ of\ words\ in\ V^u}$$

Let Φ denote

$$\frac{p_{ML}(w_i)}{sum\ of\ ML\ estimates\ of\ prob.\ of\ words\ in\ V^u}$$

We can see that in Katz's method, when the probability mass and the denominator of Φ are fixed, the larger the *ML* estimate of the probability of w_i, the higher the value of $\alpha(w_{i-1})p_{ML}(w_i)$. In some cases, this is reasonable, but not in other cases. For example, consider the bigram pattern, "on Francisco", which would never occur in any English text since it is syntactically wrong. According to Eq. (6), we have.

$$p(Francisco \mid on) = \alpha(on) \times p_{ML}(Francisco)/C(on)$$

since the phrase "San Francisco" is very common in English text. Hence, $p_{ML}(Francisco)$ will be high. Then $p(Francisco \mid on)$ will be high too. This result is unreasonable in English language.

One shortcoming of Katz smoothing is only $p_{ML}(w_i)$ is taken into account in distributing the probability mass. $p_{ML}(w_i)$ only provides information about back word w_i as an isolated entity. To some extent, the probability mass is blindly distributed among unseen bigram patterns in Katz's method. The identity of w_i has much to do with its preceding word w_{i-1}. Hence, it would be helpful to take the interdependent information between w_{i-1} and w_i into account in distributing the probability mass. When the direct information between w_{i-1} and w_i is unavailable ($C(w_{i-1}w_i) = 0$), it is natural to try to find less specific information between w_{i-1} and w_i to assist determining the value of $p(w_i \mid w_{i-1})$.

3.3 Kneser-Ney smoothing

It is clear that the information about the relationship between w_{i-1} and w_i would be helpful for distributing probability mass among unseen bigram patterns. Reinhard Kneser and Herman Ney[8] (1995) presented a novel scheme, which can be formulated as in Eq. (9) (given here for a bigram, Kneser-Ney's method is also applicable to higher order N-gram model).

$$p_{Kneser\text{-}Ney}(w_i \mid w_{i-1}) = \begin{cases} \dfrac{C(w_{i-1}w_i) - D}{C(w_{i-1})}, & \text{if } C(w_{i-1}w_i) > 0, \\ \alpha(w_{i-1}) \dfrac{||\{v \mid C(vw_i) > 0\}||}{\sum_w ||\{v \mid C(vw) > 0\}||}, & \text{if } C(w_{i-1}w_i) = 0 \end{cases} \tag{9}$$

$$\alpha(w_{i-1}) = \frac{1 - \sum_{w_i: C(w_{i-1}w_i)>0} p_{Kn-Ne}(w_i \mid w_{i-1})}{\sum_{w_i: C(w_{i-1}w_i)=0} \left[\dfrac{||\{v \mid C(vw_i) > 0\}||}{\sum_w ||\{v \mid C(vw) > 0\}||} \right]}$$

Again, α is a normalization constant that guarantees the probability sums to 1. Compared with Katz smoothing method, Kneser and Neys smoothing uses a simpler discounting scheme rather than computing the discounts using Good-Turing smoothing technique, a single discount, D, is used. $D = n_1/(n_1 + 2n_2)$, n_1, and n_2 are the frequencies of frequencies of 1 and 2 respectively. The numerator, in the case of $C(w_{i-1}w_i) = 0$, is the number of words which can precede w_i to form bigrams, while the denominator is the total number of kinds of possible bigram patterns contained in training data.

According to Kneser-Ney's method, the word "Francisco" only appears in rather few contexts. That is, "Francisco" as a back word, is able to form bigram patterns with few preceding words. Such as "San". Thus, $\dfrac{||\{v \mid C(v\text{“}Francisco\text{”}) > 0\}||}{\sum_w ||\{v \mid C(vw) > 0\}||}$ will be small.

Accordingly, $p(Francisco \mid on)$ will be small. This is reasonable. On the contrary, the word, "Tuesday" may appear in many contexts. That is, it is able to form many different bigram patterns with different preceding words. Therefore, $\frac{|\{v \mid C(v\text{"}Tuesfay\text{"})>0\}|}{\sum_w |\{v \mid C(vw)>0\}|}$ will be relatively high. Kneser-Ney smoothing performs better than Katz. This has been proven by Goodman[5](1999). Our study in this paper also affirms this.

3.4 Special character bigram information based Katz smoothing(Scb-Katz)

Kneser and Ney introduced a special relationship information between w_{i-1} and w_i to assist the distribution of probability mass. He find that the number of words preceding a given word to form various seen bigram patterns depends on the identity of the given word. He distributed the probability mass according to this property of the rear word of the N-gram pattern. Inspired by their success, we propose two kinds of useful information between w_{i-1} and w_i to assist the distribution work. They have proven effective in our experiments and will be discussed in the current and next subsections.

Unlike English, in which each word is composed of letters, Chinese words are made up of characters. A Chinese word may contain one or more characters. Two-character words are frequently used while words containing more than 5 characters rarely used. In this study, we only investigate words containing one to five characters. Every Chinese character has different meaning and function. Thus, the probability of one word followed by another word is different. Based on this point, researchers have established Chinese character bigram language model before, which reflects how likely two Chinese characters might appear consecutively.

Motivated by the character bigram model, we suggest a variant of it, which is able to characterize how likely two Chinese characters at special position can appear consecutively. This special character bigram is to investigate the concurrence probability of the rear character of the front word and the head character of the back word of the bigram pattern. In other words, we are to investigate the concurrence probability of two characters, which appear at special positions in two consecutive words.

We use this information together with Φ to decide how large a portion of probability mass would be assigned to $C_{Katz}(w_{i-1}w_i)$ when $C(w_{i-1}w_w)=0$. Let $p(HZ_{wi} \mid RZ_{w_{i-1}})$ be the probability of the head character of back word conditioning on the rear character of front word. Eq. (6) can be rewritten as Eq. (10).

$$C_{Katz}(w_{i-1}w_i) = \begin{cases} r, & \text{if } r>k \\ d_r r, & \text{if } 0<r\leqslant k \\ probability\ mass\ [\gamma\Phi + \\ (1-\gamma)p(HZ_{w_i} \mid RZ_{w_{i-1}})], & \text{if } r=0 \end{cases} \tag{10}$$

where γ is a coefficient , which will be determined empirically.

3.5 Word class bigram information based Katz smoothing (WordClass-Katz)

Apart from the special character bigram information which can reflect the relationship between w_{i-1} and w_i at a less specific level, we also discover that the word class conditional probability $p(Class(w_i) \mid Class(w_{i-1}))$ can also be used as less specific information between w_{i-1} and w_i, where $Class(w_i)$ and $Class(w_{i-1})$ denote the word class of w_i and the word class of w_{i-1} respectively. All words falling in the same word class usually have similar syntactical attributes. Those attributes decide how likely they appear before or after other words belonging to different or same word class. Therefore, it is reasonable to consider the relationship between two word classes as an approximated relationship between w_{i-1} and w_i. Finally we use knowledge about $p(Class(w_i) \mid Class(w_{i-1}))$ and $p_{ML}(w_i)$ to guide the distribution of probability mass. This can be formulated as Eq. (11).

$$C_{Katz}(w_{i-1}w_i) = \begin{cases} r, & \text{if } r>k \\ d_r r, & \text{if } 0<r\leqslant k \\ probability\ mass[\gamma\Phi + (1-\gamma)\cdot \\ \quad p(Class(w_i) \mid Class(w_{i-1})], & \text{if } r=0 \end{cases} \tag{11}$$

In our study, we create 91 word classes based on words' functions in a sentence and their parts of speech. First, we generally divide words in V into 46 major classes. Those categories are actually grammatical categories according to syntactical behavior of the

words. Some of the 46 major classes can be divided into finer subclasses according to some important attributes. The finer the classification, the more accurate the information about $p(Class(w_i) \mid Class(w_{i-1}))$ would be. According to the same attributes, the word class verb can be divided into finer subclasses.

3.6 Improved Kneser-Ney smoothing

The performance of Kneser-Ney smoothing can also be improved by using the special word class bigram information or character bigram information. The additional information was integrated into the Kneser-Ney smoothing according to the formula (12) and (13) respectively. We call them WordClass-Kneser-Ney model and Scb-Kneser-Ney model.

$$p_{Kneser\text{-}Ney}(w_i \mid w_{i-1}) = \begin{cases} \dfrac{C(w_{i-1}w_i) - D}{C(w_{i-1})}, & \text{if } C(w_{i-1}w_i) > 0, \\ \alpha(w_{i-1})\left[\dfrac{||\{v \mid C(vw_i) > 0\}||}{\sum_w ||\{v \mid C(vw) > 0\}||}\beta + (1-\beta) \cdot p(Class(w_i) \mid Class(w_{i-1}))\right], & \text{if } C(w_{i-1}w_i) = 0 \end{cases} \tag{12}$$

$$p_{Kneser\text{-}Ney}(w_i \mid w_{i-1}) = \begin{cases} \dfrac{C(w_{i-1}w_i) - D}{C(w_{i-1})}, & \text{if } C(w_{i-1}w_i) > 0, \\ \alpha(w_{i-1})\left[\dfrac{||\{v \mid C(vw_i) > 0\}||}{\sum_w ||\{v \mid C(vw) > 0\}||}\beta + (1-\beta) \cdot p(HZ_{w_i} \mid RZ_{w_{i-1}})\right], & \text{if } C(w_{i-1}w_i) = 0 \end{cases} \tag{13}$$

4 Experiments

In this section, we describe our experimental results about performances of the following language models: character bigram model, Katz's word bigram model, Kneser-Ney's word model, Scb-Katz model, WordClass-Katz model, WordClass-Kneser-Ney model and Scb-Kneser-Ney model.

In our experiments, the size of vocabulary is around 32 000 words. These words may be constituted of one to five Chinese characters, *i.e.*, the longest word has 5 Chinese characters. The size of word bigram model is about 6.5 M bytes. We also get a segmented and tagged training data of 1 million characters from an online database. We trained the word class bigram model $p(Class(w_i) \mid Class(w_{i-1}))$ on it.

We use a handwritten Chinese character recognition system to test the performance of the discussed language models. First we feed a sentence into the recognizer. The recognizer produces 10 best character candidates for each character in the sentence. These character candidates will form a character candidate lattice with each character candidate occupying a cell. Based on the character candidate lattice, we produce a word candidate lattice by identifying all potential words contained in the character lattice. We apply dynamic programming algorithm and character-based language model information to the character lattice or dynamic programming algorithm and word-based language model information to word lattice to find a best path through the lattice as a postprocessing result. The postprocessing result is usually better than the original recognizer's result. In Table 1, we show the performance difference between basic Katz smoothing and improved Katz smoothing; in Table 2, we show the performance difference between basic Kneser-Ney smoothing and improved Kneser-Ney smoothing. The larger the performance difference, the better the effect of our improvements.

Table 1 Effect of our improvements on Katz smoothing

	Word Class-Katz LM	Scb-Katz LM	Kneser-Nay LM
Recognition rate	89.1500	88.9914	89.0100
Basic Katz LM	88.6895	88.6895	88.6895
Difference	0.4605	0.3019	0.3205

Table 2 Effect of our improvements on Katz smoothing

	WordClass-Kneser-Ney LM	Scb-Kneser-Ney LM
Recognition rate	89.20	89.10
Basic Kneser-Ney LM	89.01	89.01
Difference	0.19	0.09

Our tests were conducted over a text of 100 000 characters, about 11 000 sentence, which differ from any training data. The original 1-best recognition rate is 82.56% while the 10-best recognition rate is 91.66%. In Table 3 and Table 4, with character bigram model as baseline, we list, respectively, the performances of Katz smoothing family and Kneser-Ney smoothing family in terms of recognition rate. The character bigram LM is used as baseline.

Table 3 Performances of Katz smoothing based language models

	Character bigram model(%)	Basic Katz model(%)	WordClass-Katz model (%)	Scb-Katz model(%)
Postprocessing result	85.2155	88.6895	89.1500	88.9914
Original rate	82.5630	82.5630	82.5630	82.5630
Difference	2.6525	6.1265	6.5870	6.4284

Table 4 Performances of Kneser-Ney smoothing based language models

	Character bigram model(%)	Kneser-Ney model(%)	WordClass-Kneser-Ney (%)	Scb-Kneser-Ney(%)
Postprocessing result	85.2155	89.0100	89.2000	89.1000
Original rate	82.5630	82.5630	82.5630	82.5630
Difference	2.6525	6.4470	6.6370	6.5370

When γ in Eq. (10) and Eq. (11) is 0.66 the improved Katz model can achieve optimal performance. When β in Eq. (12) and Eq. (13) is 0.60, the improved Kneser-Ney model can achieve optimal performance.

The improvement by our suggestions is not very remarkable. We think the main reason is that we are trying to make further improvement from the results of other improved measurements. It is more difficult to make a further progress from an improved platform than from an original platform. Furthermore, we implemented the word classification according to a Chinese information dictionary in the subsection 3.5. Currently we only have the demo version of the this dictionary, we are not able to do finer word classification.

The recognition error rate of original system is 100% - 82.56% = 17.43%. The recognition error rate of the basic Katz smoothing is 100% - 88.68% = 13.31%. A 23.665% lower recognition error rate is achieved.

In a Microsoft technique reports(2001)[6], Goodman reported a similar improvement effect by Katz and Kneser-Ney smoothing on an English speech recognition system. In their experiments, test was conducted on 600 utterances using a speech recognition system, the size of their vocabulary is 58 546. The 100-best recognition error rate is 5.2 %. The 1-best recognition rate is 10.1%. The improved recognition rate is around 9.76% by Katz model and 9.58% by Kneser-Ney model. A 3.4% and a 5.12% recognition error rate reduction are made respectively. It seems that our method can achieve a better result. However, the author stressed that all parameters in their system was optimized to minimize the specification perplexity instead of recognition error rate. Furthermore, they were using a state-of-the-art speech recognizer. Hence, there is little room for LM to improve. But they admitted that the perplexity is proportional to recognition rate. Usually, the improvement produced by a language model is small in terms of recognition rate though larger in terms of perplexity. However, recognition rate is a more objective and meaningful specification. Thus, in our study, we use recognition rate as the metric to gauge the performance of the language models.

5 Conclusion

In this paper, we investigate the performances of some word bigram language models with a Chinese character recognition system. For word bigram language models, the common problem is the data sparseness. For backing-off smoothing technique, some pieces of probabilities are first spared from overestimated bigrams. These probability pieces conglomerate to be probability mass. Then, the probability mass is distributed among unseen bigrams according to a kind of probability distribution. In Katz's distribution scheme only probability information of w_i is used, but no interdependent information between current word and its historical word(s). In our modified version of Katz smoothing scheme, a kind of special character bigram information and class bigram information is used respectively. The experimental results proved that the two kinds of information is helpful to Katz smoothing technique.

References

[1] Bahl, Lalit R, Brown P F, *et al*. A tree-based statistical language model for natural language speech recognition [J]. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 1989, **37**, 1001 - 1008.

[2] Brown, Peter F, Vincent J, *et al*. Class-based *n*-gram

models of natural language [J]. *Computational Linguistics*, 1992, **18**(4): 467-479.

[3] Good I J. The population frequencies of species and the estimation of population parameters [J]. *Biometrika*, 1953, **40**: 237-264.

[4] Katz S M. Estimation of probabilities from sparse data for the language model component of a speech recognizer [J]. *IEEE trans. on Acoustics, Speech, and Signal Processing*, 1987, **35**(3): 400-401.

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

[6] Goodman J T. A bit of progress in language modeling extended version [R]. MSR-TR-2001-72 Technique Report, Microsoft Research, Microsoft Corporation, 2001.

[7] Chen S F, Goodman J. An empirical study of smoothing techniques for language modeling [R]. Technique Report, TR-10-98, Computer Science Group, Harvard University, 1998.

[8] Kneser R, Ney H. Improved backing-off for *M*-gram language modeling [A]. *In Proceeding of the IEEE International Conference on Acoustics, Speech and Signal Processing* [C]. 1995, **1**.

[9] Kneser R, Ney H. Improved clustering techniques for class-based statistical language modeling [J]. *In Eurospeech*, 1993, **2**: 973-976.

10 Kneser R, Ney H. Statistical language modeling using a variable context length [J]. *In ICSLP-96*, 1996, **1**: 494-497.

11 Goodman J, Gao J. Language model size reduction by pruning and clustering [J]. *In ICSLP*, 2000.

12 Iyer R, Ostendorf M, Rohlicek J R. Modeling long distance dependence in language: topic mixtures versus dynamic cache models [J]. *IEEE Trans. on Acoustics, Speech and Audio Processing*, 1999, **7**(1): 30-39.

13 Zhang R, Black E, Finch A, Sagisaka Y. Integrating detailed information into a language model [J]. *In ICASSP-2000*, 2000: 1595-1598.

14 Church K W, Gale W A. A comparison of the enhanced Good Turing and deleted estimation methods for estimating probability of English bigrams [J]. *Computer Speech and Language*, 1991, **5**(5): 19-54.

15 Charniak E. Immediate-head parsing for language models [J]. *In ACL-01*, 2001: 116-123.

16 Siu M, Ostendof M. Variable *n*-grams and extension for conversational speech language modeling [J]. *IEEE Trans. on Speech and Audio Processing*, 2001, (1): 63-75.

17 Manning C, Schutze H. Fundamentals of Statistical Nuatural Language Processing [M]. The MIT Press, 1999.

(*Executive editor YAO Yue-Yuan*)