A LANGUAGE MODEL IN A LARGE-VOCABULARY SPEECH RECOGNITION SYSTEM

Jin-ming ZHAN, Xiaolong MOU, Shuqing LI, Ditang FANG
Speech Lab, Dept of Computer Science, Tsinghua University, Beijing, China
Email: fangdt@mail.tsinghua.edu.cn
Telephone: +86-10-6278 4141

ABSTRACT

A language model for large-vocabulary speech recognition system is introduced in this paper. The Turing's probability estimation of n-grams is discussed in detail. The search strategy used in the language model is also described. In the end, some experiment results are presented to show the effectiveness of the estimation and search techniques we have employed.

1. INTRODUCTION

A large-vocabulary speech recognition system should have two primary parts: the acoustic model and the language model. The acoustic model turns the utterance into a list of candidates, which are exported to the language model as its input. The language model, with some pre-calculated results, determines the most possible word sequence. Their relation can be described by the following graph:

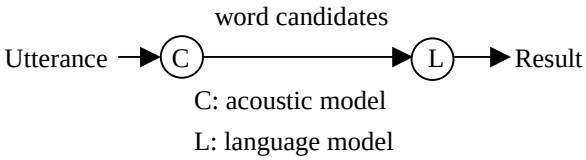

Fig-1. The diagram of a speech recognition system.

Suppose the input of the acoustic model, or the spoken words, are A, and the output of the acoustic model and the input of the language model, or possible word strings, are *w*, the system's task is to find the most possible word string w^* which satisfies

$$\begin{aligned} & P(w^* \mid A) \\ &= \arg\max_w P(w \mid A) \\ &= \arg\max_w \frac{P(A \mid w)P(w)}{P(A)} \end{aligned} \tag{1.1}$$

The second step is derived from the Bayes rule. *P(A)* is constant, which we may just ignore. *P(A|w)* is the probability calculated by the acoustic model. *P(w)* is what the language model takes care of and what we will mainly focus here.

In our system, we use a vocabulary of 24,713 words, including punctuation. The input to the acoustic model consists of words separated by recognizable pauses. The language model we use is a trigram model. To obtain reliable trigram results, the model is trained by a text corpus of 50 million words. The Turing's estimation is used to estimate the unseen trigrams, a method which we will fully discuss later.

2. THE N-GRAM MODEL

The n-gram language model is based on the following assumption: the *n*th word is only related to its preceding *n-1* words. Therefore, the probability estimation of the language model *P(w)* can be rewritten as $P(w_n|w_1,...,w_{n-1})$. For a sentence with *N* words, given the candidates $w_1,...,w_N$ from the acoustic model, the probability of the whole sentence is calculated as:

$$P(w) = \sum_{i=1}^{N} P(w_i \mid w_{i-n+1}, \ldots, w_{i-1}) \tag{2.1}$$

2.1. The basic n-gram model

For large enough text corpora, the probability of $P(w_n|w_1,...,w_{n-1})$ can be estimated from the maximum likelihood principle like this:

$$P(w_n \mid w_1,\ldots,w_{n-1}) = \frac{C(w_1,\ldots,w_n)}{C(w_1,\ldots,w_{n-1})} \quad (2.2)$$

in which $C(w_1,...,w_{n-1})$ and $C(w_1,...,w_n)$ represent the occurrence number of the word string $w_1,...,w_{n-1}$ and $w_1,...,w_n$, respectively.

In practical systems, the n is usually no more than *3*, for the reason of CPU processing ability and memory size. But for a large vocabulary of more than 10,000 words, the combination of $w_1,...,w_n$ is of great number even if $n=3$. The result is that many $C(w_1,...,w_n)$ become zero, which lead to an unreliable estimation of $P(w_n|w_1,...,w_{n-1})$. Due to the sparse training data, some combinations of $w_1,...,w_n$, which might make sense but didn't occur in the training text, are regarded as impossible, a case which may degrade the recognition rate greatly. In order to give an unseen combination a reasonable probability estimation, we employ a Turing's method.

2.2. Turing's estimation

The maximum likelihood estimation of $P(w_n \mid w_1^{n-1})$ is given in (2.2). (In the following discussion, we use w_1^n for $w_1,...,w_n$.) Now we define its Turing's estimation as

$$\tilde{P}(w_n \mid w_1^{n-1}) = \frac{k_{C(w_1^n)+1}}{k_{C(w_1^n)}} \cdot \frac{C(w_1^n)+1}{C(w_1^{n-1})} \quad (2.3)$$

Here k_r is the number of n-grams which occur in the training corpus for exactly r times. For the n-grams which occur in the training text and have the same preceding w_1^{n-1}, the sum of their maximum likelihood probability estimation is 1, i.e.

$$\sum_{w_n} P(w_n \mid w_1^{n-1}) = 1 \quad (2.4)$$

The sum of their Turing's estimation is

$$\begin{aligned}
&\sum_{w_n} \tilde{P}(w_n \mid w_1^{n-1}) \\
&= \sum_{w_n} \frac{k_{C(w_1^n)+1}}{k_{C(w_1^n)}} \frac{C(w_1^n)+1}{C(w_1^{n-1})} \\
&= \frac{1}{C(w_1^{n-1})} \sum_r k_r \cdot \frac{k_{r+1}}{k_r} \cdot (r+1) \\
&= \frac{1}{C(w_1^{n-1})} \cdot C(w_1^{n-1}) \qquad \left(\sum_r r \cdot k_r = C(w_1^{n-1}) \right) \\
&= 1
\end{aligned}$$

Therefore, for the n-grams which don't occur in the training corpus, the sum of their Turing's probability estimation is

$$\sum_{w_n: C(w_1^n)=0} \tilde{P}(w_n \mid w_1^{n-1}) = 1 - \sum_{w_n: C(w_1^n)>0} \tilde{P}(w_n \mid w_1^{n-1}) = \frac{k_1}{C(w_1^{n-1})} \quad (2.5)$$

By discounting the probability of those observed n-grams, we now have a sum of probability estimation for those n-grams which don't occur in the training corpus. Each observed n-gram has some contribution to the sum. Therefore, the probability estimation of unseen n-grams is no longer zero. The sum of all the probability estimation $\tilde{P}(w_n \mid w_1^{n-1})$ given a w_1^{n-1}, however, still equals 1.

Now that we have a sum for all the unseen n-grams, the next issue should be how to distribute this probability "mass" to the individual n-grams. Instinct tells us that $\tilde{P}(w_n \mid w_1^{n-1})$ should be somewhat proportional to $\tilde{P}(w_n|w_2^{n-1})$. We define (2.5) as $\tilde{\delta}(w_1^{n-1})$ and assume that there exists a low-level probability estimation $\tilde{P}(w_n|w_2^{n-1})$. So the estimation of $\tilde{P}(w_n \mid w_1^{n-1})$ for an unseen w_1^n can be written as:

$$\tilde{P}(w_n|w_1^{n-1}) = \gamma(w_1^{n-1})\tilde{P}(w_n|w_2^{n-1}) \quad (2.6)$$

where

$$\gamma(w_1^{n-1}) = \frac{\tilde{\delta}(w_1^{n-1})}{\sum_{w_1^n : c(w_1^n)=0} \tilde{P}(w_n | w_2^{n-1})} \quad (2.7)$$

When $c(w_1^{n-1}) = 0$, we define

$$\tilde{P}(w_n | w_1^{n-1}) = \tilde{P}(w_n | w_2^{n-1}) \quad (2.8)$$

Now, we have finished the definition of Turing's estimation. For the observed n-grams, their probability is estimated by (2.3). For the unseen ones, their probability is estimated by (2.6) or (2.8), depending on whether w_1^{n-1} occurs.

3. SEARCH STRATEGY

In our system, we use a Trigram model in which *n*=3. Suppose we have a sentence of *N* words. For each word W_i, there are *m* candidates $w_{i_1}, w_{i_2}, \ldots, w_{i_m}$. Since we are using a Trigram model, the search node becomes a "word pair". To search for the optimal word sequence, the maximum path to each word pair must be saved in each step. Therefore the time complexity for searching the maximum path is $O(m^3N)$. Obviously, *m* is a major factor which determines the search time. However, *m* is not determined by the language model, but the acoustic model, which always yields 10-20 candidates for each spoken word. By cubing this number, it becomes some thousand. On another aspect, the search process will not stop until it comes to the last word, or the *N*th word. We identify the end of a sentence by punctuation. The punctuation is specially treated in our system so that it can be recognized more quickly and accurately. So most of the time, we can tell exactly the end of each sentence, or the value of *N*. The larger the *N* is, i.e. the longer the sentence is, the longer the search time is. In order to make this time shorter, we employ two strategies here.

In the first one, we introduce a window technique. The window covers the candidates of *s* words. The search strategy described above is used on these candidates. After the calculation in the window completes, the window shifts to the right by *t* words. So the system gives *t* words as the results of each search in the window.

The second one is to decrease the influence of *m* to the search time. Instead of saving the maximum paths to each word pair in a single search step, we only keep the first 10 optimal ones. Experiment results show that with this simplification, the speed improves drastically.

It is easy to see that both the two strategies have made some comprise between the accuracy and the speed. The search result may not be the optimal one, but the error rate is within tolerance.

4. RESULTS

The Turing's estimation described above discounts every observed n-gram to give a contribution to the sum of the unobserved ones. But for some n-grams which occur more than 6 times in the training corpus, we consider them as reliable enough and don't want them to be discounted. So for these n-grams, we still use their maximum likelihood estimation. For those occurring less than 6 times in the training corpus, we rewrite (2.3) as:

$$\tilde{P}(w_n | w_1^{n-1}) = \frac{k_1 \cdot k_{C(w_1^n)+1}[C(w_1^n)+1] - 7k_{C(w_1^n)} C(w_1^n) k_7}{k_{C(w_1^n)} C(w_1^{n-1})(k_1 - 7k_7)} \quad (4.1)$$

The performance of a language model is evaluated by its perplexity, defined as:

$$H = -\frac{1}{n} \log P(w_1^n) \quad (4.2)$$

For the Trigram model in our system, equation (2.1), which is used to calculate the overall probability estimation of a whole sentence, becomes:

$$P(w) = P(w_1)P(w_2 | w_1) \sum_{i=1}^{N-2} P(w_{i+2} | w_i w_{i+1}) \quad (4.3)$$

So the perplexity of the Trigram model is

$$H = -\frac{1}{N-2} \sum_{i=1}^{N-2} P(w_{i+2} | w_i w_{i+1}) \quad (4.4)$$

We used 100 sentences to calculate the perplexity of our

language model. The result is shown below.

Model	1*	2**
Bigram	121	117
Trigram	90	88

Table 1. The perplexity of language models.

*: The basic n-gram model.

**: The revised n-gram model using Turing's estimation.

We can learn from the above table that the perplexity of the language model decreases as we increase the *n* and use the Turing's estimation.

Three experiments were performed to test the accuracy of the language model, using the combinations of the methods we have discussed. First, we tested how much the Turing's estimation improves the system's performance. 200 sentences were selected as the sentences to recognize. For each word in a sentence, we randomly selected 19 words plus the correct one as the candidates for that word. Table 2 shows the results of these two estimation methods. From the results presented in Table 2, we can see a great improvement in the accuracy rate.

	Accuracy Rate
Turing	93%
Maximum Likelihood	75%

Table 2. The accuracy rates of two estimation.

In the second experiment, we compare the accuracy rate of two methods. First, we include the correct one in each word's candidate list (Method 1). Then, in order to test the robustness of this algorithm, one word was made to have no correct candidate in its candidate list (Method 2). The results are shown below.

	Accuracy Rate
Method 1	93%
Method 2	88%

Table 3. The accuracy rates of two methods.

The results show that this algorithm depends much on the output from the acoustic model. If the correct one is not included in a word's candidate list, several of its preceding and following words will be influenced. So we are thinking of using some prediction algorithms to fortell the next word based on the first few recognized words. Further details will be presented in future papers.

The third experiment was designed to test the accuracy rates of different search strategies. For Method 1 in the last experiment, we tested the accuracy rates of the full search and the revised one mentioned in section 3. Here are the results.

	Accuracy Rate
Full Search	93%
Revised Search	90%

Table 4. The accuracy rates of two searches.

Therefore, it's a good deal to sacrifice the performance a little for a much higher speed.

From the above discussions and results, we can see that by using the Turing's estimation and the revised search strategy in our system, a good result can be obtained.

5. REFERENCES

[1] V. Gupta, M. Lennig and P. Mermelstein, *A language model for very large-vocabulary speech recognition*, Computer Speech and Language (1992) 6, 331-344

[2] Kenneth W. Church and William A. Gale, *A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams*, Computer Speech and Language (1991) 5, 19-54

[3] Slava M. Katz, *Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer*, IEEE transactions on acoustics, speech and signal processing, Vol. ASSP-35, No. 3, March 1987

[4] Arthur Nadas, *On Turing's Formula for Word Probabilities*, IEEE transactions on acoustics, speech and signal processing, Vol. ASSP-33, No. 6, December 1985