INTEGRATING LONG-DISTANCE LANGUAGE MODELING TO PHONEME-TO-TEXT CONVERSION

Tai-Hsuan Ho[1], Kae-Cherng Yang[2], Juei-Sung Lin[1], Lin-Shan Lee[123]
[1]Department of Computer Science and Information Engineering, National Taiwan University,
[2]Department of Electrical Engineering, National Taiwan University,
[3]Institute of Information Science, Academia Sinica,
Taipei, Taiwan, R.O.C.
email : tai@speech.ee.ntu.edu.tw, Tel. : 886-2-369-2535

Abstract

This paper presents a phoneme-to-text conversion system for Chinese language using long-distance language modeling. First of all, we employ extended bigrams (Huang 1993) of window size *d* to capture the long-distance dependent relations in Chinese language, in which *d* bigram tables are estimated independently from the training data for distance 1 to *d*. Each bigram table is associated with a mixture weight, which can be optimized based on the held-out data using deleted interpolation algorithm (Ney 1994). The system then performs the tree-trellis search (Soong 1991) to generate N-best sentence hypotheses, and integrates these extended bigram probabilities at sentence level. In our experiments, we generate 200 best sentence hypotheses and the integration of long-distance bigram reduces the error rate by about 11% as compared with word bigram language model only. Secondly, to reduce the number of parameters, we merge the extended bigram tables from distance 2 to *d* to form a single long-distance bigram table, disregarding the influence caused by different distances. Since the model complexity is significantly reduced, we derive a very efficient stack decoding algorithm for the integration of this augmented long-distance information. Experiments show that the error rate remains the same as that of *d* extended bigrams using N-best search algorithm, while the search efficiency is significantly improved.

1. Introduction

It has been studied intensively for many years to find good approaches to input Chinese characters, for which standard keyboard is not well suited. Technologies including Speech Recognition, Hand-Writing Recognition, Optical Character Recognition (OCR) have been adopted for this task, aiming to provide users with very efficient and natural ways to input Chinese characters. However, these systems are not prevailing mainly because special hardware

is required or the technologies are not mature enough to be accepted by the users. Keyboard input methods are still the most widely used ones. There are tens of different methods for the input of Chinese using keyboard. In this paper, we confine ourselves to deriving good models and search algorithms for the phoneme-to-text conversion task, converting the input tonal syllable sequence into Chinese characters. These syllables are composed of sequences of phonemes from the standard phoneme set (ㄅ, ㄆ, ㄇ, ㄈ, etc.) and 5 tones. Each phoneme and tone is associated with a key stroke. In Chinese language, a large portion of syllables have tens of homonym characters, and some of them even have more than a hundred. The goal of this system is therefore to automatically pick the right one from the homonym set as accurate as possible.

There are a lot of related researches being explored in this topic. Among them, 漢音 (Kuo 1995) and GOING (Hsu 1994) are probably the most noticeable ones. 漢音 system uses a rule-based and statistic-based hybrid method to this problem. Morphological rules, word length heuristics, and syntactic and semantic connection tables are applied to enhance the accuracy. In GOING system, Semantic Pattern Matching based approach is adopted, which requires linguistic experts to derive a lot of templates at different levels for Chinese language. Since some templates can model Chinese language at phrase or sentence level, GOING system has been one of the systems that successfully handle the long-distance dependent relations in Chinese language. Instead of deriving costly templates and rules, here we propose a completely statistical approach to the integration of long-distance language modeling. We first apply the current language processing technology of Golden Mandarin (III) (Wang 1997) speech recognizer to solve the phonetic Chinese input problem as our baseline system, and then integrate long-distance Markov language model to enhance the accuracy.

Markov source language model has been widely adopted and proven very effective in many language processing tasks, such as speech recognition and machine translation. Given a word sequence $W_{1,n} = w_1, w_2, ..., w_n$, the language model probability for this sentence is

$$\Pr(W_{1,n}) = \prod_{i=1}^{n} \Pr(w_i | W_{1,i-1}) \qquad (1)$$

To robustly estimate the probabilities and reduce the number of parameters, n-gram language model assume the probability for the current word w_i depends only on its previous n-1 words $W_{i-n+1,i-1}$. Equation (1) thus becomes

$$\Pr(W_{1,n}) \cong \prod_{i=1}^{n} \Pr(w_i | W_{i-n+1,i-1}) \qquad (2)$$

In Golden Mandarin (III), we use n equals to 2, namely bigram, which turns out to be a good tradeoff between model complexity and performance. However, bigram fails to model the dependency for words with distance longer than 2, and this kind of dependent relations can be found quite frequently in many cases. Some examples are listed below. In these example , each sentence has been segmented properly into the corresponding word sequence, and words underlined are assumed mutually dependent.

Example 1: 洗 了 一 個 舒服 的 澡 (take a comfortable bath)

Example 2: 依 這種 法律 規定 (according to this regulation)

Example 3 : 一 隻 可愛 的 小 花 貓 (a lovely little spotted cat)

Given the phoneme sequences of these examples, n-gram with small n fails to convert them to their corresponding Chinese characters correctly. In the first example, Golden Mandarin (III) will convert the last word to another homonym "早 (morning)", since it has higher probability than the correct one "澡 (bath)". It will never be correctly converted unless the system takes into consideration the verb at the begin of the sentence "洗 (take)". Similarly in example 2 and 3, "依 (according to)" will be converted to "一 (one)" and "隻 (indefinite article for animal)" replaced by "支 (indefinite article for equipment)" if only bigram is applied.

To remedy these errors, our approach first use extended bigrams of window size d to capture the long-distance dependency. Unlike bigram language model that can be integrated efficiently by dynamic programming algorithm, long-distance model can only be integrated at sentence level by using algorithms such as N-best search. In this task, large N is necessary to avoid missing the global optimum. We use N=200 in our experiments, and the integration of extended bigrams reduces the error rate by about 11% as compared with word bigram language model only. Secondly, we merge these extended bigram tables from distance 2 to d to reduce the number of parameters, and derive a very efficient stack decoder to integrate this augmented long-distance model. Experiments show that the performance is significantly improved without degrading the accuracy. On a Pentium-Pro 200 MHz machine, the processing speed can be as high as 60 characters per second after fully optimizing the implementation.

The rest of this paper is organized as follows. Next section describes our baseline system : language processing module of Golden Mandarin (III) using word bigram. Section 3 describes

the training and estimation procedure of the extended bigrams adopted in our system, and also briefly describes the N-best search algorithm for the integration of these bigram tables. In Section 4, we smooth the long-distance information by merging the extended bigram tables from distance 2 to *d*, and derive a stack decoder for the integration of the merged long-distance model. In section 5 we show our experimental results for the algorithms and models described in previous sections, and conclude our remarks in section 6.

2. Baseline System : Language Processor of Golden Mandarin (III)

Our baseline system is the language processor of Golden Mandarin (III) speech recognizer. The input to this module is a syllable lattice, with which many confusing syllable candidates are included for each speech segment. For the phoneme-to-text conversion task, only one syllable is present at each segment. The architecture for the phoneme-to-text conversion baseline system is shown in Figure 1. Given the input phoneme sequence, a word lattice is first constructed by exhaustively looking up the lexicon for all possible word hypotheses, in which all possible transcriptions for this input phoneme sequence are encompassed. Figure 2 gives an example of the partial word lattice for the input phoneme sequence

"ㄒㄧㄢˇ ㄌㄜ˙ ㄧˊ ㄍㄜ˙ ㄕㄨ ㄈㄨˊ ㄉㄜ˙ ㄗㄠˇ"

A word lattice is a Direct Acyclic Graph (DAG) in which each node represents a word, and each arc represents a possible word transition. Each arc is associated with a bigram transition probability which can be estimated automatically from the training data. Word lattice is a very compact representation for all possible sentence hypotheses. Each path from the sentence start to the sentence end is a possible transcription. In this example, the best transcription for this syllable sequence is "洗 了 一 個 舒服 的 澡" (take a comfortable bath), corresponding to the path connected by the thick lines in Figure 2.

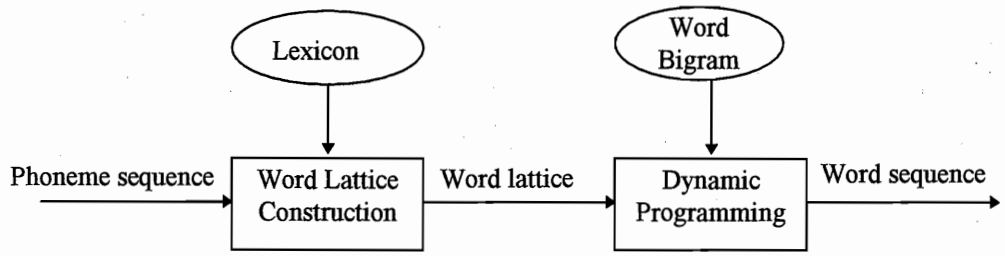

Figure 1. The architecture of the baseline system

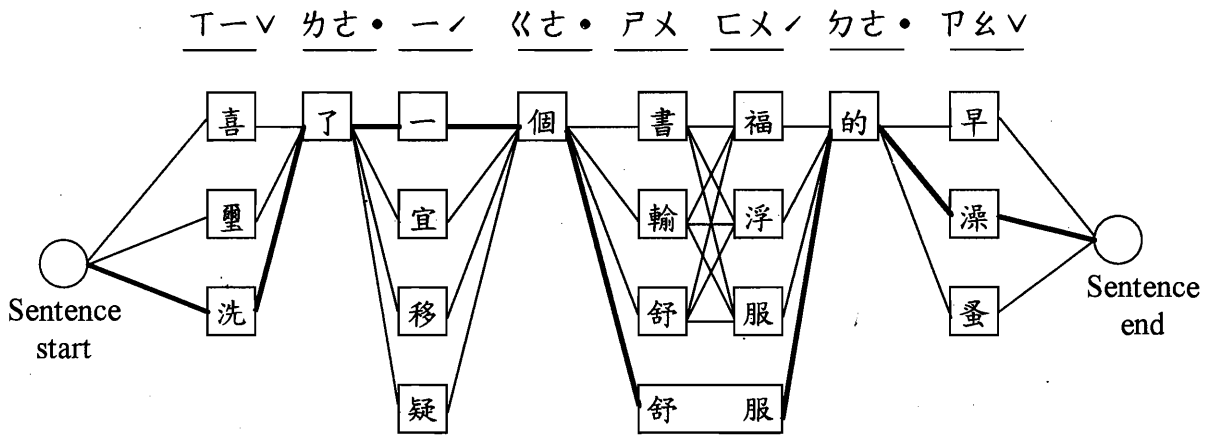

Figure 2. An example of a partial word lattice and corresponding the best path

The language model of the baseline system is word bigram. Given a path in the lattice with word sequence $W_{1,n} = w_1, w_2, ..., w_n$, the word bigram probability for this word seqauence is

$$\Pr(W_{1,n}) = \Pr(w_1) \times \Pr(w_2 | w_1) \times ... \Pr(w_n | w_{n-1}) \quad (3)$$

The bigram probability can be estimated using Maximal Likelihood Approach to maximize the total probability of the training data. However, for testing data, it may result in zero probability for word pairs not present in the training data, hence unreasonable zero probability for the whole testing data. This happens when the available training data is sparse, and can be solved by back-off to unigrams for unseen bigrams. Our back-off scheme is adopted from BBN's approach (Placeway 1993). The bigram probability for word pair $w_j w_i$ is

$$\Pr(w_i | w_j) = \tilde{\Pr}(w_i | w_j) + \bar{\Pr}(w_i | w_j) \quad (4)$$

and

$$\tilde{\Pr}(w_i | w_j) = \frac{Count(w_j w_i)}{Count(w_j) + Branch(w_j)} \qquad \text{for } Count(w_j w_i) > 0 \quad (5)$$

$$\bar{\Pr}(w_i | w_j) = \frac{Branch(w_j)}{Count(w_j) + Branch(w_j)} \times \Pr(w_i) \qquad \text{otherwise} \quad (6)$$

where *Branch(w)* is the branch factor, number of distinct succeeding words, of *w* found in the

training data. In equation (5), denominator is enlarged by its branch factor, resulting in a smaller probability for the trained word pairs. This lost probability is then distributed to all possible word pairs as their back-off bigram probabilities, and the distribution is proportional to their unigram probabilities $\Pr(w_i)$ of the current words, as described in equation (6). The back-off coefficient at equation (6) is roughly proportional to the branch factor $Branch(w_j)$ of the previous word w_j. There is a physical meaning for this back-off coefficient. If the branch factor of the previous word w_j is large, then the probability that word w_j can transit to w_i should be large, too, even though this transition was not found in the training data. If this branch factor is small, then the untrained transition probability should be small as well, since it is quite determined to transit to some specific words only.

After constructing the word lattice, the search engine will extract the best path maximize equation (3). The search engine of the baseline system contains a forward dynamic programming pass and a backtrace pass. In the forward pass, for each node w the system will compute the best partial path score from sentence start to w, α_w, using the following equation.

$$\alpha_w = \max_u \{\alpha_u + \log \Pr(w|u)\} \qquad (7)$$

where u are all words in the lattice immediately preceding w. For each word w, the best preceding word $\tilde{u}$ is recorded. In this way, the best path can be obtained by tracing $\tilde{u}$ from the sentence end all the way to sentence start in the backward pass.

3. Extended Bigrams and N-best Search Algorithm

In a traditional stochastic language model, the current word is predicted based on the preceding word (bigram) or the preceding n-1 words (n-gram). This is because most of the relevant syntactic information can reasonably be expected to lie in the immediate past. But some information, syntactic as well as semantic, may still exist in the more distant past, though use of n-gram with large n will increase the number of free parameters exponentially. To reduce the number of free parameters and maintain the modeling capacity, we use a set of extended bigrams (Huang 1993) for different distances to approximate the Markov source language model probability, as shown in the following equation :

$$\Pr(w_i | W_{1,i-1}) = \sum_{k=1}^{d} \lambda_k \Pr_k(w_i | w_{i-k}) \quad (8)$$

where $\Pr_k(w_i | w_{i-k})$ is the probability that predicts word w_i based on the word w_{i-k}. In our system, we use d=5, assuming the long-distance dependency can be ignored for distance greater than 5. In equation (8), each distance is associated with a mixture weight λ_k, which can be optimized based on the held-out data using deleted interpolation algorithm (Ney 1994) as described in the following equation.

$$\lambda_k^{(t+1)} = \frac{1}{N} \sum_{i=1}^{N} \frac{\lambda_k^{(t)} \Pr_k(w_i | w_{i-k})}{\sum_{k=1}^{d} \lambda_k^{(t)} \Pr_k(w_i | w_{i-k})} \quad (9)$$

where N is the size of the held-out data. The above equation re-estimate the next set of mixture weights $\lambda_k^{(t+1)}$ based on the current one $\lambda_k^{(t)}$. The iteration continues until the change of all the $\lambda_k^{(t)}$ can be neglected. Also, to fully utilize the available training data, we subdivide them into 5 parts, and re-estimate λ_k in a leave-one-out manner (Duda 1973).

The search algorithm follows the tree-trellis N-best search paradigm (Soong 1991) which contains 2 stages. In the first stage, the system efficiently generates N-best sentence hypotheses using word bigram only, and in the second stage re-scores these hypotheses using more complex models such as extended bigrams, as shown in Figure 3. The one with the highest sentence probability is then hypothesized as the conversion result. The search algorithm for generating N-best sentence hypotheses using word bigram from the word lattice is briefly summarized in the following steps :

1. Perform forward dynamic programming algorithm and compute α_w using equation (7) for every word w in the word lattice
2. Push initial hypothesis which contains sentence end node only into the stack
3. Pop the best hypothesis h from the stack. Hypothesis h is started with word w
4. Go To 9 if h is a complete sentence and N-best sentences have been generated
5. Go To 3 if h a complete sentence and N-best sentences have not been generated
6. For each word v precedes w, extend hypothesis h to word v, create a new hypothesis h', and compute the score for h'

   $$\beta(h') = \beta(h) + \log \Pr(w|v) \quad (10)$$

$$score(h') = \alpha_v + \beta(h') \quad (11)$$

7. Push all the new hypotheses *h'* into the stack
8. Go to step 3
9. Return the N-best sentence hypotheses

To ensure the global optimum is included in the N-best sentence hypotheses, large N is usually required. In our experiments , we observe N=200 is sufficient enough for our test data, and N larger than 200 does not yield higher accuracy. The final system with extended bigram results in 95.3% conversion accuracy, representing a 11% error reduction as compared with word bigram only.

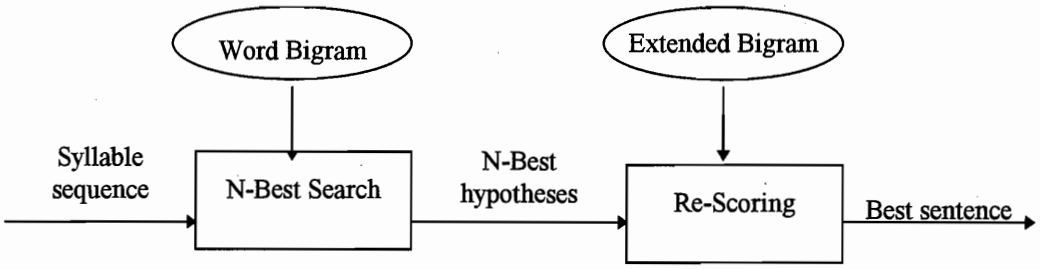

Figure 3. N-Best Search paradigm for the phoneme-to-text conversion using extended bigrams

4. Merged Long-distance Bigram and Stack Decoder

In the previous section we successfully integrate the extended bigrams into our phoneme-to-text conversion system. However, the proposed model and algorithm are not practical for real applications. The drawbacks are twofold. Firstly, it requires *d* bigram tables to store the long-distance information for extended bigrams with window size *d*. In our case *d* equals to 5, which means the storage requirement is increased by 400% but only achieves 11% error reduction. Secondly, for the N-best search algorithm, it takes a lot of efforts to generate many hypotheses and re-scores them, resulting a very inefficient system. On a Pentium-Pro 200 MHz machine, the system can only convert 22 Chinese characters per second.

To reduce the storage requirement, we merge the extended bigram tables from distance 2 to *d* to form a single long-distance bigram, diminishing the influence brought by different distances. More precisely, we assume the current word w_i, in addition to the its immediate previous word w_{i-1}, can also be predicted by any word in $W_{i-2,i-d} = w_{i-2}w_{i-3}...w_{i-d}$ with equal

possibility. In our model, the probability of word w_i given its past word sequence $W_{i-1,i-d} = w_{i-1}w_{i-2}...w_{i-d}$ is

$$\Pr(w_i | W_{i-1,i-d}) \cong (1-\lambda_{ld})\Pr(w_i | w_{i-1}) + \lambda_{ld}\Pr_{ld}(w_i | W_{i-2,i-d}) \quad (12)$$

where $\Pr(\bullet|\bullet)$ is the word bigram probability, $\Pr_{ld}(\bullet|\bullet)$ the long-distance bigram probability. The long-distance model weight λ_{ld} can, again, be optimized using equation (9). For simplicity in the search algorithm, $\Pr_{ld}(\bullet|\bullet)$ is approximated by

$$\Pr_{ld}(w_i | W_{i-2,i-d}) \cong \max_{j=i-2}^{i-d} \Pr_{ld}(w_i | w_j) \quad (13)$$

Here we assume the long-distance dependency relies on the relations of word pairs. To avoid over-smoothing the parameters by merging many bigram tables, we select only those word pairs having high mutual information within a window based on the approaches proposed by Church and Hanks in (Church 1988). This concept is similar to that of Trigger Pair language model using Maximal Entropy Approach (Rosenfeld 1996). The differences are that we are dealing with long-distance relations at sentence level instead of document level in (Rosenfeld 1996) for adaptation, and using conventional Maximum Likelihood Approach to estimate our long-distance model parameters.

To integrated this merged model, we employ a stack decoder which contains a forward dynamic programming pass and a backward stack decoding pass. Unlike N-best search in the previous section, the long-distance probability are integrated earlier in the forward pass, so as to provide a more accurate heuristics for the backward stack decoder. When performing the forward dynamic programming, each node will examine all its preceding, but not adjacent, nodes in the lattice for long-distance bigram pairs, pick the maximal one, and integrate it into the computation of α. The corresponding equation for computing α_w for word w is as follows :

$$\alpha_w = (1-\lambda_{ld}) \times \max_u [\alpha_u + \log(w|u)] + \lambda_{ld} \times \max_v \log \Pr_{ld}(w|v) \quad (14)$$

where u are words preceding and adjacent to w and v are words preceding but not adjacent to w in the lattice. This step seems very time-consuming, since for every node we need to examine all its preceding nodes in the lattice. Fortunately, because the number of long-distance bigram pairs

are quite limited after the selection based on mutual information, we can use a very simple bookkeeping technique to efficiently identify all possible pairs having long-distance relations in the lattice.

In the backward pass, the stack decoder use these α in each nodes for heuristic functions. Since the computation of α in equation (14) are always over-estimated, the stack decoder is A* admissible, and the first decoded sentence is guaranteed to be the global optimum. The search algorithm is summarized in the following steps :

1. Perform dynamic programming algorithm and compute α using equation (14)
2. Push initial hypothesis containing only sentence end into the stack
3. Pop the best hypothesis h from stack. Hypothesis h is started with word w
4. If h is a complete path, go to step 8
5. For each word v immediately precedes w, extend hypothesis h to word v, create a new hypothesis h', and compute $\beta(h')$ using equation (1), (12) and (13)

   $$score(h') = \alpha_v + \beta(h') \qquad (15)$$

6. Push all the new hypotheses h' into the stack
7. Go to step 3
8. Return the decoded sentence

In our system, we use DEAP data structure (Horowitz 1991) to implement the stack, which gives O(logN) complexity for the push and pop operations. The stack size is limited, and hypotheses ranked lower than this limit is truncated permanently. Experiments show that the accuracy is almost the same as that of extended bigram and N-best search, while both storage requirement and performance are significantly improved. On a Pentium-Pro 200 MHz machine, this system can convert about 60 Chinese characters per second in average.

5. Experimental Results

In our experiments, the language model training data are all newspapers provided by CKIP of Academia Sinica, containing about 12 million words. The lexicon contains about 42K words in which all Chinese characters are included as single-character words. Before training the language model, sentences from the training data are first segmented into word sequences in a data-driven, iterative manner. The algorithm is initialized by using a longest match strategy, and at each iteration aligns sentences against word entries in the lexicon based on bigram criterion,

aiming to reduce the perplexity for the training data. After several iterations , the lowest perplexity we obtained for these training data is 156. The test data contains about 110K Chinese characters, 80% of them are from newspaper, and the rest 20% are from other domains including prose, tourism, sports, art and ecology, etc. The perplexity for the test data is 382. Using our baseline system with word bigram only, we obtain 94.7% character accuracy for the test data, and the conversion speed is about 126 Chinese characters per second.

For extended bigrams, we use d=5 and assume the effect of long-distance dependency can be ignored for distance greater than 5. After optimized by deleted interpolation algorithm, the mixture weight corresponding to each distance is shown in the following table :

λ_1	λ_2	λ_3	λ_4	λ_5
0.68	0.16	0.08	0.03	0.05

Table 1. Mixture weight corresponding to different distances

In this table we can observe the weight for d=1 takes the lion's share, indicating a large portion of context dependency have been modeled by bigram. The mixture weight for d=2 is decent. Modeling context dependency for distance longer than 1 should be beneficial. The experimental results using extended bigrams and N-best search are shown in Table 2, where the accuracy and conversion speed with respect to different numbers of hypotheses are listed. In this table, the accuracy is saturated when the number of hypotheses exceeds 200. The best result obtained is 95.3%, representing a 11% error reduction as compared with 94.7% of the baseline system. The speed for this system is about 22 characters per second.

N	1	10	20	30	50	100	**200**	300	400
Accuracy (%)	94.7	94.8	95.1	95.1	95.1	95.2	**95.3**	95.3	95.3
Speed (char./sec)	126	112	94	75	49	38	**22**	12	8

Table 2. Experimental results for the extended bigrams : Character accuracy and conversion speed with respect to different N in N-best search

For the merged long-distance bigram, we truncate long-distance pairs based on mutual information criterion. Using stack decoder with stack size 16K, the accuracy obtained given different numbers of parameters are shown in Table 3.

Number of long-distance pairs	10K	20K	50K	**100K**	200K	400K	800K
Accuracy (%)	94.9	94.9	95.1	**95.3**	95.3	95.2	94.8

Table 3. Experimental results for the merged long-distance bigram : Character accuracy with respect to different numbers of long-distance bigram pairs

It can be found that the accuracy is increased as the number of long-distance pairs proceeds from 10K to 100K, but is degraded when more than 200K long-distance pairs are included in the language model. Merging these extended bigram tables can result in inaccurate parameter estimation, and therefore introduce some noise information. However, truncation based on mutual information helps to remove these noisy information, and turns out to be a good industrial tradeoff for the consideration of efficiency. In this experiment, the best result obtained is 95.3%, which is the same as that of extended bigram and N-best search. The search efficiency, however, is significantly improved. We achieve 60 characters per second for the conversion speed.

6. Conclusion

Modeling long-distance dependency reduces the error rate reasonably for the Chinese phoneme-to-text conversion task. In this paper, we present two models for the long-distance context dependency, extended bigrams and merged long-distance bigram. Two approaches are also presented for the integration of these long-distance models, N-best search and stack decoder. Tree-trellis based N-best search can integrate extended bigram language models, but need to generate many sentence hypotheses to cover the global optimum. Besides, huge storage is required for the extended bigram models. In order to reduce the storage requirement, we merge all the extended bigram tables from distance 2 to d to form a single long-distance bigram table, and derive a very efficient stack decoder for the integration of this merged long-distance model. Experiments show that the integration of long-distance information reduces error rate by 11%. Using stack decoder and merged long-distance bigrams, the system can convert 60 Chinese characters per second.

Reference

Church, K. W. and Hanks, P. "Word Association Norms, Mutual Information, and Lexicography", ACL 1988, pp. 76-83

Duda, R. O. and Hart, P. E. "Pattern Classification and Scene Analysis" Wiley, New York, 1973

Horowitz and Sahni, "Fundamentals of Data Structure in PASCAL" third edition. Computer Science Press, pp. 534-541, 1991

Hsu, W. L., "Chinese Parsing in a Phoneme-to-Character Conversion System Based on Semantic Pattern Matching", Computer Processing of Chinese and Oriental Languages, Vol. 8, No. 2, December 1994, pp. 227-236

Huang, X. D. et al, "An Overview of the SPHINX-II Speech Recognition System", Proceeding of the ARPA Human Language Technology Workshop 1993. Published as *Human Language Technology*, pp. 81-86, Morgan Kaufmann

Kuo, J. J. "Phonetic-Input-to-Character Conversion System for Chinese Using Syntactic Connection Table and Semantic Distance", ICCPOL 1995, pp. 286-292

Ney, H. et al, "On Structuring Probabilistic Dependences in Stochastic Language Modeling". Computer Speech and Language (1994) 8, pp. 1-38

Placeway, P. et al, "The Estimation of Powerful Language Models from Small and Large Corpora", ICASSP 1993, pp. II-33 to II-36

Rosenfeld, R., "A Maximum Entropy Approach to Adaptive Statistical Language Modeling", Computer Speech and Language (1996) 10. pp.187-228

Soong F. and Huang, E. F. "A Tree-Trellis Based Fast Search for Finding the N Best Sentence Hypotheses in Continuous Speech Recognition:, ICASSP 1991, pp. 705-708

Wang, H. M. et al, "Complete Recognition of Continuous Mandarin Speech for Chinese Language with Very Large Vocabulary Using Limited Training Data", IEEE trans. On Speech and Audio Processing, vol. 5, NO. 2, March 1997