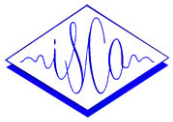

International Symposium on Chinese Spoken Language Processing (ISCSLP 2000)
Fragrant Hill Hotel, Beijing
October 13-15, 2000

IMPROVED STRATEGIES FOR INTELLIGENT SENTENCE INPUT METHOD ENGINE SYSTEM

Ling JIN , Genqing WU , Fang ZHENG , Wenhu WU

Center of Speech Technology, State Key Laboratory of Intelligent Technology and Systems,

Department of Computer Science & Technology, Tsinghua University, Beijing, 100084

{jinl, wgq, fzheng}@sp.cs.tsinghua.edu.cn

ABSTRACT

This paper describes a Chinese keyboard intelligent full-sentence input method system based on tri-gram language model. In this system, we use efficient algorithms to reduce the size of the language model, accelerate the search and enhance the accuracy. The n-gram model is presented in a novel structure, which shrinks the model and enables it with look-ahead and buffer techniques to reduce the times of visiting the disk to fetch the n-gram unit and to adapt the language model to user domain quickly. Besides that, we have designed an efficient dynamic programming algorithm to segment input alphabetic sequence into syllabic cells; thereby it can be fit for different input ways.

1. INTRODUCTION

The goal of our Input Method Engine (IME) system is to translate a letter string into a Chinese sentence, where the letter string can be a pinyin string, an initial string or a mixed pinyin and initial string without any space in it, where both the initial (without final followed) and pinyin can be regarded as a syllable delegate. The first step of our IME system is segmentation, which is, in other words, to insert necessary spaces in the boundaries of syllable delegates. Different segmentation methods result in different putative syllable delegates, so an effective searching algorithm must be adopted for segmentation. We use an effective dynamic programming algorithm for segmentation. To maximize the number of letters involved in segmentation and to minimize the number of outgoing syllable delegates, this algorithm can achieve the best result by scanning letters only one pass.

In other IME systems, when entering a sentence, users must choose from many word candidates. To avoid this disturbing work, we use an algorithm based on maximum likelihood by integrating n-gram language model. An Initial\Final tree structure based word lexicon is used in this algorithm. Because the synchronous syllable decoding algorithm sorts the paths by partial scores, when comparing two kinds of paths together in one stack, where one kind is the paths whose searching state reach leaf nodes (word boundary) of the word-lexicon and the other for the contrary circumstance, some potential paths will be pruned. We use double-stacks to store each kind of paths respectively to avoid this problem. And look-ahead cache techniques are also used to accelerate the search marvelously.

To make the language model's size smaller, we choose a relatively smaller glossary containing the most frequently used words. To avoid the problem of data sparseness, we uplift the discount threshold and normalize the counts of the words in our modified back-off language model. In order to make it adaptable for different kinds of people, we make two N-gram models: one is the System model, and the other is the User model. Combining user input and System model properly can adapt the whole model to a specific field quickly without bringing too much instability. In addition, the User model is much smaller than the System model, which enables the flexibility in adaptation.

2. DYNAMIC PROGRAMMING FOR SYLLABLE SEGMENTATION

2.1 Why We Need Dynamic Programming Algorithm?

Usually, we seldom use disjunctive letter among syllables when entering a whole sentence by IME. Sometimes the result of segmentation is ambiguous and it will make recognition accuracy low. For example: "fangan", we can decode it into either "fan 'gan "(反感) or "fang' an" (方案). Therefore it's necessary to do some segmentation after user input some letters. Because our IME has to support many input methods, such as initial input, pinyin input … so the results of segmentation are syllable delegates. Using general searching algorithms isn't effectual enough. Here we use dynamic programming for segmentation.

2.2 Dynamic Programming Algorithm

In the dynamic programming algorithm, we need an optimal principle. After referring to many IME systems, we define such an optimal principle for calculating and searching.

Here suppose we want to segment M letters (input by user) into N syllable delegates. First, we should make more and more letters inputted by the user to take part in segmentation. Second, try to find smaller N while in the condition in the first principle is still met.

The advantages of using these two optimal principles are:

- It can take more letters into consideration in the segmentation.

- It's fit for the input habit of users.
- Smaller N can make searching more effectively.

$S = S_1 S_2 \ldots S_n$: User's input stream where S_i can be a lowercase letter or a symbol "'". The disjunctive symbol "'" is a compulsive disjunctive mark. Array *D*(*I*) : minimal disjunctive parts composed by first I letters. Array *C*(*I*) : The first position of the syllable *D*(*I*).

We can describe the algorithm as follow:

- *D(0)=0; D(1)=1; C(1)=0;*
- *D*(*I*)= min{ *D*(*I-K*)+1, where *K*+1 letters No. *I-K*+1 through *I* can make up a syllable delegate, 1<=*K*<=6, *K*<=*I*}
- *C(I)= C(arg D(I)).*

Given one *I*, if a reasonable *K* cannot be found, we make *D(I)=D(I-1) +1, C(I)=I;* If there exists more than one *K* for each *I*, we choose the smallest K. Because the longest syllable has 6 letters (such as "zhuang "、 "shuang "), therefore we make *K* no bigger than 6.

2.3 Outline Of Dynamic Programming

The expatiation of our dynamic programming algorithm is as follows.

1. Initialization. Let *D[0]=1, C[0]=0* and define maximal syllable numbers : *MAXS= 50*, make start position: *nowseat=1*;
2. *C[nowseat]=-1 , D[nowseat]=MAXS+1;*
3. If letter *S[nowseat]* is a disjunctive symbol "'" , let *D [nowseat] = D[nowseat-1], C[nowseat]=nowseat*, then trun to step 5; otherwise go on with step 4;
4. Find the minimal *K* base on the optimal principal, let *D [nowseat]=D[nowseat-K]+1, C[nowseat]=nowseat-K+1*;
5. *Nowseat* plus 1, if *nowseat* is smaller than the length of total input letter, then turn back to step 2; If equal then end.

3. LANGUAGE MODEL AND DECODING ALGORITHM

In our IME system, when finishing the letters-to- syllable-delegates segmentation step, the following step is to decode from these syllable delegates to the most possible Chinese words based on language model. If the scale (word number) of the language or the syllables' number is big, normal search algorithm will be infeasible. We use word lexicon tree to store word items instead of word lattice and use syllables synchronous decoding algorithm to speed up the search.

3.1 Language Model In IME System

The language model is used to decode from some syllable units $S = S_1 S_2 \ldots S_n$ into only one Chinese word sequence $W = W_1 W_2 \ldots W_m$ or Chinese characters $C = C_1 C_2 \ldots C_n$ where $W_i = C_{i_1}^{i_l}$ Actually , we want to find the maximal likelihood W* such that:

$$W^* = \arg\max_w P(W|S) = \arg\max_w P(W)P(S|W)$$

$$= \arg\max_w P(W) \prod_{i=1}^{m} P(S_{k(i)}^{l(i)} | W_i)$$

where :

$$P(S_k^l | w) = P(w | S_k^l)$$

$$= \begin{cases} 1 & w = C_k \ldots C_l, S_j \in Syll(C_j),\ j = k \ldots l \\ 0 & others \end{cases}$$

$Syll(c)$ are possible syllable set of character c.

In our IME system, we use double-stack syllables synchronous algorithm for decoding [2][4].

3.1.1 Compression of language model

The scale of language model is large and sparse. It needs to design a structure to make model smaller and more effective. Here, we develop an index file to receive these points. In considering the ease of IME adaptation, we store a unit's occurrence count of the training set in it.

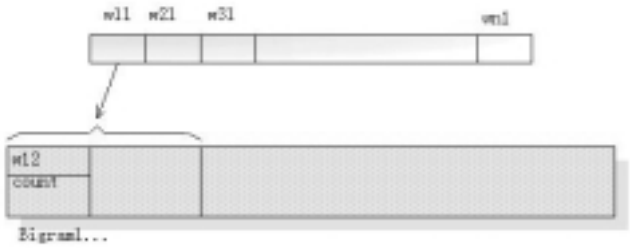

Figure 1: Bi-gram storage demonstration

In figure 1, for bi-gram, each word has one index pointer pointing to the head of this bi-gram unit's store area. Effectively, the whole bi-gram set is sorted by (word1 word2). In this way, it can seek the target unit rapidly. The situation of tri-gram is similar.

In N-gram model , many units' occurrence times is zero, a good way is to smooth them according to lower order's N-grams [1].

So we should store the smoothing coefficient and occurrence time for each unit. We make shared index because each unit's coefficient and occurrence times have the same subscript, that we cut half of the storage for index.

On the other hand, the smoothing technique can be used in the compression of the language model. If the difference between the probability $p(w_{i+2} | w_i w_{i+1})$ calculated from tri-gram model directly and the probability $p_1(w_{i+2} | w_i w_{i+1})$ estimated from lower order N-gram is small, it shows that $p(w_{i+2} | w_i w_{i+1})$ carries not much additional information and can be deleted. For simplification we assume that back-off weights do not change while compressing. This method can reduce the size of model about 10%.

3.1.2 Cache look-ahead

The probability of a given sentence ' $...w_i w_{i+1} w_{i+2} w_{i+3}....$' can be calculated as

$$P(S) = ...P(w_{i+2} \mid w_i w_{i+1}) * P(w_{i+3} \mid w_{i+1} w_{i+2})....$$
(3-1)

while the smoothed probabilities $P(w_{i+2} \mid w_i w_{i+1})$ are evaluated by one for the these three smoothing formulae:

1) if the count $C(w_i, w_{i+1}, w_{i+2})$ is non-zero (indicating (w_i, w_{i+1}) is seen in the training data), the smoothing formula is:

$$P(w_{i+2} \mid w_i w_{i+1}) = C(w_i, w_{i+1}, w_{i+2}) / C(w_i, w_{i+1})$$
(3-2)

2) if $C(w_i, w_{i+1}, w_{i+2})$ is zero while $C(w_i, w_{i+1})$ is non-zero(which ensures that the smoothing coefficient $\alpha(w_i, w_{i+1})$ exists), the smoothing formula is :

$$P(w_{i+2} \mid w_i w_{i+1}) = \alpha(w_i, w_{i+1}) * P(w_{i+2} \mid w_{i+1})$$
(3-3)

3) if neither $C(w_i, w_{i+1}, w_{i+2})$ nor $C(w_i, w_{i+1})$ is non-zero, the smoothing formula is:

$$P(w_{i+2} \mid w_i w_{i+1}) = P(w_{i+2} \mid w_{i+1})$$
(3-4)

Most of the counts $C(w_i, w_{i+1}, w_{i+2})$ are zero in the model because of the data sparseness of tri-gram language model. It is proofed in our experiment that more than 60% of smoothing tri-gram cell probabilities are calculated by (3-3), which leads more than 36% of two sequential cells probabilities to be estimated by (3-3) (in case of that the two cells are independent).

Bi-gram probabilities $P(w_{i+2} \mid w_{i+1})$ are calculated like $P(w_{i+2} \mid w_i w_{i+1})$, but most of them are directly estimated from $C(w_i, w_{i+1})$ and $C(w_i)$. If the sequential probabilities $P(w_{i+2} \mid w_i w_{i+1})$ and $P(w_{i+3} \mid w_{i+1} w_{i+2})$ are both calculated by (3-3) while the corresponding bi-gram probabilities are directly calculated from bi-gram cells and unigram cells, the data stream is shown as following:

step *I-1*:….

step *I* to calculate $P(w_{i+2} \mid w_i w_{i+1})$:

1. get $C(w_{i+1}, w_{i+2})$;
2. get $\alpha(w_i, w_{i+1})$;
3. output $\alpha(w_i, w_{i+1}) * C(w_{i+1}, w_{i+2}) / C(w_{i+1})$;

step *I+1* to calculate $P(w_{i+3} \mid w_{i+1} w_{i+2})$:

1. get $C(w_{i+2}, w_{i+3})$;
2. get $\alpha(w_{i+1}, w_{i+2})$;
3. output $\alpha(w_{i+1}, w_{i+2}) * C(w_{i+2}, w_{i+3}) / C(w_{i+2})$;

step *I+2*….

$C(w_{i+1}, w_{i+2})$ is accessed in step *I*, and $\alpha(w_{i+1}, w_{i+2})$ is accessed in step *I+1*, the two cells have the same suffix (w_{i+1}, w_{i+2}), they should be stored together , so that when $C(w_{i+1}, w_{i+2})$ is fetched in step *I* , $\alpha(w_{i+1}, w_{i+2})$ can be fetched in advance and buffered, which avoids repeating useless search in step *I+1* and enhance the search efficiency.

Analysis on the training data shows that cells in sentence will occur again soon. In order to improve the performance of search, buffer techniques are taken to store the cells' probabilities that have been searched in the past. Our experiments show great improvement in search efficiency[5].

3.2 Improved Word Lexicon Tree

The Word Search Tree (WST) was designed to reflect the relations among all these in-vocabulary words so that the redundancy for both the vocabulary storage is reduced [2].

In such a lexicon tree the traveling direction is always from the parent node to its child node(s), so it can be stored in a linear data structure, i.e., an array of nodes. It takes only several seconds to establish a WST for 51, 200 words using a well-defined algorithm.

In order for the system to adapt the user's input easily, we should memorize user's input. If a word (or a phrase) isn't in our word lexicon, we should add this word (phrase) into word lexicon. Add word continually will make lexicon tree grow with more nodes. We use indexed arrays, instead of stack , to store word lexicon, so inserting nodes will cause much memory movement. It'll make IME respond user's input delay. Therefore, we improved the structure of the word lexicon(as figure 2):

- Let each node of the tree stand for one initial, one final or one Chinese word instead of one syllable.
- Cut the original lexicon into several small lexicon trees. We classified the whole model into 23 classes by indexing the first letter of one initial or one final. Each class makes one lexicon tree.

4. ADAPTATION

In order to make it adaptable for different kinds of people, we take measures as follows: The N-gram model consists of two sub-models, one is the system model, and the other is the user model [3]. The system model is achieved from the training corpus, which is a general model for all fields, and the user's input combined with the system model forms the user model. Combining user input's and the system model properly can make it adapt the whole model to a specific field quickly without bringing too much instability. In addition, the user model is much smaller than the system model, which enables the flexibility in adaptation.

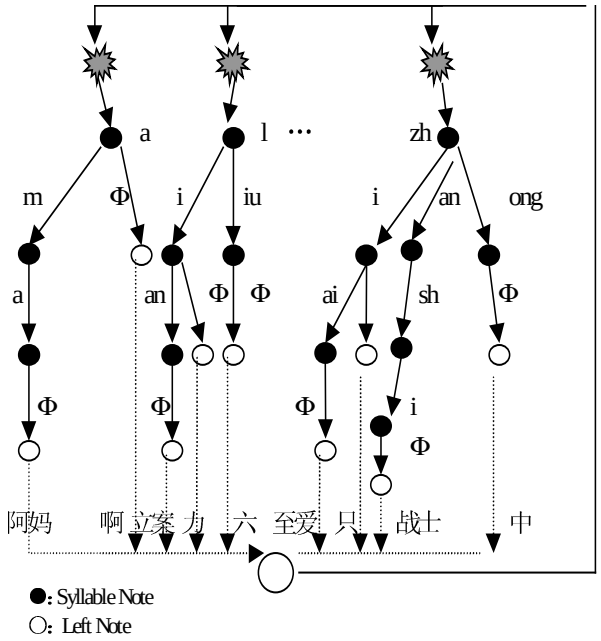

Figure 2: Improved lexicon tree

5. EXPERIMENT RESULTS

5.1 Inserting A Word Into Lexicon

Improved structure of word lexicon: we make several small lexicons instead of one whole lexicon so as to decrease the memory movements.

	Old Lexicon	**Improved Lexicon**
Number of all notes	24197	40617
Node movements when inserting a new word (average condition)	12098.5	882.9

Table 1: Node movement

It can be seen that when inserting a word, the movement of memories is decreased to about 7% of the original.

5.2 Memory Reading Of Language Model Unit

We test a cache area which has total 100 units, hash seeking address , testing in trained texts. The result is:

Tri hit rate	$\alpha(w_i, w_{i+1}, w_{i+2})$ **hit rate**	**Using cache/ Before using cache (seconds)**
4.95%	31.9%	2167/1284

Table 2: Experiment about new storage structure and cache technique.

Although, the hit rate of Tri is not very high, but the hit rate of $\alpha(w_i, w_{i+1}, w_{i+2})$ is pretty high. It can be seen from Table 2 that the speed can increase about 70%.

6. CONCLUSION

In our IME system, we use an efficient dynamic programming algorithm to segment input alphabetic sequence into syllabic cells, thereby it can be fit for different input ways. Improved word lexicon can make memory movement decreased. Look-ahead cache techniques are also used to accelerate the search marvelously. Two N-gram models are used for adaptation. By using these strategies, the performance of IME is improved effectively.

7. REFERENCES

[1] Mou X.-L., Zhan J.-M., Zheng F. and Wu W.-H "The back-off algorithm based N-gram language model, " *5th National Conference on Man-Machine Speech Communication (NCMMSC-98)*, 206-209, 1998 (In Chinese)

[2] Ho.T-H. Yang , K-C. Huang, K-H "Improved search strategy for large vocabulary continuous mandarin speech recognition" *Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing, ICASSP98, 1998.*

[3] Wolfgang Reichl "Language model Adaptation Using Minimum Discription Information, " *EuropSpeech 99 , v4 , pp 1791-1794.*

[4] Zheng.F, Song Z.J, Xu M.X., etc "EasyTalk: A large-vocabulary speaker-independent Chinese dictation machine", *Proceedings of 6th European Conference on Speech Communication and Techniques, EUROSPEECH'99. Budapest, Hungary, 1999, v2, pp.819-822*.

[5] S.M.Katz "Estimation of Probabilities from Sparse Data for the Language model Component of a Speech Recognizer". In *IEEE Transactions on Acoustics, Speech, and Signal Processing 35(3)*, pp. 400-401, 1987.