A SYLLABLE-SYNCHRONOUS NETWORK SEARCH ALGORITHM FOR WORD DECODING IN CHINESE SPEECH RECOGNITION

Fang Zheng

Speech Laboratory, Department of Computer Science and Technology
Tsinghua University, Beijing, 100084, P.R. China
fzheng@sp.cs.tsinghua.edu.cn

ABSTRACT

The Chinese language is syllabic in nature with frequent homonym phenomena and severe word boundary uncertainty problem. This makes the Chinese continuous speech recognition (CSR) slightly difficult. In order to solve these problems, a Chinese syllable-synchronous network search (SSNS) algorithm is proposed. Together with the vocabulary word search tree and the N-gram based language model, the syllable-synchronous network search algorithm gives a good solution to the Chinese syllable-to-word conversion. In addition, this algorithm is a good method for the accent Chinese speech recognition. The experimental results have showed that the SSNS algorithm can achieve a good overall continuous Chinese speech recognition system performance.

1. INTRODUCTION

Normally a large-vocabulary CSR system has two primary parts: the acoustic model and the language model. Suppose the input of the acoustic model, or the utterance, is *A*, and the output of the acoustic model which is also the input of the language model, or possible word strings, is *W*, the system's task is to find the most likely word string W^* which satisfies

$$W^* = \arg\max_W P(W|A) = \arg\max_W P(A|W)P(W) \quad (1)$$

This equation is derived from the Bayesian rule and $P(A)$ is constant and may well be ignored. $P(A|W)$ is the conditional probability of the utterance on the specific word string given by the acoustic model while $P(W)$ is the word string probability given by the language model.

But for the Chinese CSR, this would be different. It is known that Chinese is a syllabic language. A Chinese sentence is a string of Chinese words (*CW*). A *CW* consists of one or several Chinese characters (*CC*), and mostly each *CC* of a *CW* with definite meaning corresponds to a unique Chinese syllable (*CS*) in pronunciation. The following three main reasons go against the use of Equation (1), i.e., taking *CW*s as the common units for both acoustic modeling and language modeling as in western languages. Here what we mean by "common units" is the "units" of both the output of the acoustic-processing stage and the input of the language-processing stage.

Firstly, the homonym and homograph phenomena are common. There are just about 400 toneless *CS*s and about 1,300 toned *CS*s but more than 6,700 frequently used *CC*s in the Chinese language. Actually, *CS*s are the pronunciations of *CC*s. On an average, 17 *CC*s share one toneless *CS* and 5 *CC*s share one toned *CS*. Furthermore, a *CC* may have several different *CS*s in different *CW* contexts. In this situation, if *CW*s are chosen to be the speech recognition units or the acoustic frame-synchronous network search algorithm [1] or Viterbi decoding algorithm [4] is performed based on the *CW*s, there will be much redundancy in searching paths.

Secondly, though *CW*s are basic semantic units the word boundaries are hard to determine in a given sentence. The first reason is that a multiple-character *CW* can be regarded as a catenation of smaller *CW*s recursively. For example, the *CW* "计算机" can be explained to be a whole word (computer), a two-word string ("计算 + 机" with the meaning of computing machine or computer) or a three-word string ("计+算+机" with the meaning of computing/computing machine or computer). This situation causes no problem because no semantic conflict arises due to the determination of word boundaries, it just increases the word segmentation complexity. The second reason is that the semantic conflict may be caused by the uncertainty of the word boundaries. For example, the meaning of *CW* "美国会" can be "the American Congress" (美 + 国会) or "the America will (do something)" (美国 + 会) or "the meaningless" (美 + 国 + 会), depending on the context. Mostly, any *CW* can be a string of single-*CC* words or other combinations, and the meanings may be different or the same with different word boundaries. This is a problem of Chinese word segmentation, or so-called "word decoding". Currently the commonly used "maximal length matching" or "longest word first" algorithm can be used to segment a definite *CS* or *CC* string into *CW* string with the aid of a lexicon with the word frequencies. Still the word boundary problem will enlarge the redundancy in acoustic search space if the speech recognition is based on the *CW*s.

Thirdly, for real-world applications there are many kinds of accents in the Chinese language even for standard Chinese. For each accent, there exists a set of syllable mappings between this accent and the standard Chinese. For example, for a Hong Kong person, the "zhi" is pronounced as "ji". One solution to this issue is to model the mapping in the acoustic stage. This also overloads the acoustic state decoding.

To solve the above problems, an efficient algorithm is proposed in this paper. The basic idea for such a CSR system is as follows: (i) the conversion of the spoken utterance into *CS* string candidates; (ii) the conversion of the *CS* string candidates into most likely *CW* string based on the N-gram language model and the syllable-synchronous network search algorithm as proposed. The idea aims at taking syllables instead of words as the common

units and changing the Equation (1) into (2), where we define $C(W)$ as the catenation of all the syllable strings corresponding to each *CW* in the word string *W*.

$$\begin{aligned} W^* &= \underset{W}{argmax}\ P(A \mid W)P(W) \\ &= \underset{W}{argmax}\ P(A \mid S)P(W)\big|_{S=C(W)} \\ &= \underset{\substack{W \\ S=C(W)}}{argmax}\ P(A \mid S)P(W) \end{aligned} \tag{2}$$

Equation (2) gives us an good idea. The approach to find a maximal likelihood word string or sentence W^* can be divided into two stages: (1) decoding the input utterance *A* into a list of syllable string candidates *S* 's with probabilities $P(A|S)$'s; (2) finding the maximal likelihood sentence W^* under the condition $S = C(W)$. That's the basic principle of our proposed method for CSR.

2. ALGORITHM FOR WORD DECODING

In this section, the details are given for the word-decoding algorithm.

2.1 The Basic Procedure for CSR

As mentioned in Section 1, we propose a novel procedure for CSR, which differs from the word-based two-layer decoding procedure. The basic procedure is as follows.

- Given a specific utterance *A*, a syllable string candidate list is generated using such traditional speech recognition technologies as frame-synchronous network search algorithm or Viterbi decoding algorithm. To be brief, denote this stage by $A \rightarrow \{S_k \mid 1 \le k \le K\}$ where S_k is k^{th} syllable string candidate.
- $\{S_k \mid 1 \le k \le K\}$ is turned into word string candidate list $\{W \mid C(W) = S_k, 1 \le k \le K\}$. Because of the word boundary problem, each S_k will correspond to a word string set. In this stage, the proposed syllable-synchronous network search (SSNS) algorithm is used based on the morphology for the given vocabulary. For efficiency purpose, this stage will preserve only *N*-best partial word strings for each synchronous step, the score of each partial word string is given by the language model.

Details are given in the following.

2.2 Word Search Tree – The Use of Morphology

A given *CC* string may have different corresponding *CS* string (i.e. pronunciation), but a *CW* with definite semantic meaning has a unique *CS* string. To clarify, *CWs* with different meanings are treated as different *CW*s in our CSR systems, under this assumption, any word in this vocabulary has a unique *CS* string (pronunciation). Hereafter, we regard a *CW*'s *CS* string as its unique property entry, and any *L*-character *CW* can be represented by both its unique *CC* string $W = \{c_1, \cdots, c_L\}$ and its unique *CS* string $W = \{s_1, \cdots, s_L\}$, here we use "=" for simplification.

The Word Search Tree (WST) [5] was designed to reflect the relations among all these in-vocabulary words so that the redundancy for both the vocabulary storage and the acoustic searching consumption in this tree are reduced.

In this tree, we define

- *Root Node (RNode)* as the topmost node. It is a virtual node just containing pointers to its child nodes. The *RNode* has no parent node.
- *Syllable Node (SNode)* as a node that describes a certain syllable of a word. It contains information of the syllable and pointers to its child nodes.
- *Syllable Node Related String (SNR-String)* as a string of sequential syllables contained in the corresponding syllable nodes covered sequentially by the route travelling from the root node to the current syllable node in the tree.
- *Leaf Node (LNode)* as the bottommost node. It just contains information of the word whose corresponding *CS* string is exactly the same as its parent node's *SNR-String*. An *LNode* has no child node. During the word decoding, reaching an *LNode* causes an extended search from the *RNode*.

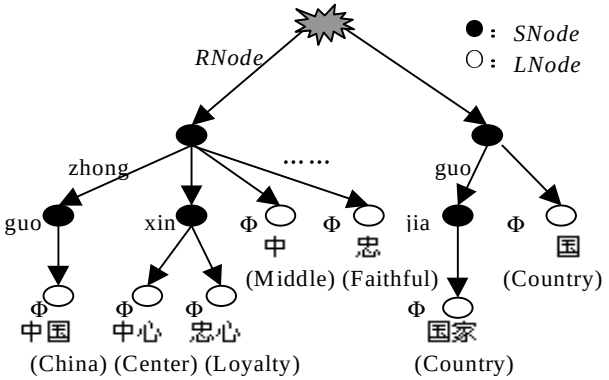

Figure 1. A partial Word Search Tree (WST)

So for any *CWs* $W_i = \{s_{i1}, \cdots, s_{iL_i}\}(i = 1,2,\cdots)$ where L_i is the length of W_i in character/syllable, if *k* is the maximal value for which $\{s_{i1}, \cdots, s_{ik}\}$ are the same for all these words, they would share the same *SNR-String* of length *k* and the same travelling route in the tree. All $s_{i,k+1}$'s, if any, are stored in brother nodes.

Thus the tree based on the morphology can be built according to the above assumption. We call this tree a word search tree (WST). By using the WST, the word-decoding procedure becomes very simple. Figure 1 shows part of a WST. By travelling through this tree, any word segmentation for a given word string can be easily covered, for example, "中国" can be covered by travelling along Route "*RNode*→zhong→guo→ Φ(中国)" (as a single word) or along Route "*RNode*→zhong→Φ(中)→*RNode*→guo→Φ(国)" (as a catenation of two words).

An actual WST is more complicated, for the vocabulary is very large and contains many relatively long Chinese words. In such a tree the travelling 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 27,000 words using a well-defined algorithm. Once the WST is available, it can be used in the CSR system and need not be re-established until the user modifies the vocabulary.

2.3 The N-Gram Based Language Modeling

When turning a Chinese syllable string $S = \{s_1, \cdots, s_L\}$ into a Chinese word string, there would be many candidates due to the uncertainty of the Chinese word boundaries. Among these word string candidates $\{W | C(W) = S\}$, there should be only one correct word string. We take the word string with most likelihood score (MLS) as the final result. Assume a word string is $W = \{w_1, \cdots, w_N\} \stackrel{def}{=} w_1^N$, its MLS is defined as the probability of the word string, which is simplified as

$$P(W) \approx P(w_1)P(w_2|w_1)\prod_{n=3}^{N} P(w_n|w_{n-2}, w_{n-1}) \quad (3)$$

This is the well-known tri-gram model. In the Chinese language, there are about 20~30K commonly used words. For such a large vocabulary, the probability matrix is quite sparse due to the lack of training data. Some tri-grams which might make sense but do not occur in the training data are regarded as impossible, a case which may degrade the recognition rate greatly. In order to give the unseen word combinations reasonable probability estimation, we employ a Turing's method [3] and modify it to be more practical one [2]. The modification of Turning's method greatly reduces the perplexity of language model.

2.4 Syllable-Synchronous Network Search (SSNS)

In this section, we will describe the SSNS algorithm in details. For convenience, we will define a data structure named as the search path to remember the instantaneous matching information between the already-processed partial syllable string and its corresponding partial word string. Mainly, each search path contains the following fields:

- *CurNode* – the pointer to the current node in the WST;
- *PWordString* – The partial word string already decoded and *PWordStringLength* – its length in word; and
- *LLScore*: the accumulated log likelihood N-gram score of the partial word string.

We assume the input is only one *CS* string $S = \{s_1, \cdots, s_L\}$, the SSNS algorithm can be described in pseudo-codes as follows.

1. Initialization, only one path starting at the *RNode* with 0-length *PWordString* and 0-valued *LLScore*.
2. For $l^{th} (1 \le l \le L)$ Chinese Syllable s_l in this string, and for each search path,
   - 2.1 Check all the child nodes of the current node that this path stops at.
     - 2.1.1 If the child node is an *SNode* and the corresponding syllable is exactly s_l, split out one path from the original one, advance the new path to this child node, and insert it; or
     - 2.1.2 If the child node is an *LNode*, split from the original path a new one and remember the corresponding word into the partial decoded word string and accumulate the *LLScore*. Search in the *RNode*'s child nodes for s_l, if founded, advance the new path to this node and insert it.
     - 2.1.3 Get rid of the original path.
     - 2.1.4 Go to 2.1 to check the next path.
   - 2.2 If the number of the new generated paths exceeds a predefined number, get rid of some least possible paths according to the *LLScore* values of the path. The criterion can be based on the *N*-best rule or a confidence measure.
   - 2.3 Go to 2 for the next syllable.
3. Sort all the paths that stop at *LNodes* in descending order according to the *LLScore* values.
4. The word string in the top 1 path is the word-decoding result.

The above description gives an outline of the SSNS algorithm. As a matter of fact, the output of the acoustic model is a parallel list of *CS* strings. In this case, the above program flow is still suitable after a little bit modification. Firstly, make all these strings the same length by appending space syllable(s) to those shorter strings. Secondly, if the s_l of a string in Step 2 is a space syllable, leave all the corresponding paths unchanged. Thirdly, enrich Step 2 by checking l^{th} syllables in all these parallel syllable strings.

2.5 A Prediction Function for Forward Matching

Because of a certain reason, we often do not save all possible paths in each step the syllable string goes forward syllable by syllable. This may cause a problem. After all syllables in the string are processed, you cannot find a single path stopping at an *LNode*. This is mostly caused by the disturbance from many impossible paths. A Prediction Function is designed to prune all these impossible paths as early as possible. In SSNS, before inserting a path for which the syllable in the WST node and the input syllable match with each other, a prediction function is called. This function tries to forward matching the rest route in the WST and the rest of the input syllable string, the path is inserted only if the function returns true. Actually this function does not try to match all the rest of the input syllable string. For example, if the attempting matching reaches an *LNode* in the WST, it definitely means the forward matching is okay.

3. THE SOLUTION TO ACCENT ISSUE

In Chinese there are many kinds of accents. The utterances are often with regular accents even if the people themselves tend to speak in standard Chinese (known as Mandarin) due to their birth and living places. In a certain accent, some spoken syllables are mapped into different syllables in the Mandarin syllable set.

Normally people like to handle this problem in the acoustic-processing stage that results in the redundancy of search paths. By using the SSNS algorithm, we give a simple and efficient solution in the language-processing stage.

3.1 Fuzzy Syllable Pair and Fuzzy Syllable Set

By using the knowledge of regional accents, we can list several accent-syllable to Mandarin-syllable mapping pairs for any kind of accent. For example, Hong Kong people often pronounce "zhi" into "ji", "chi" into "qi", "shi" into "xi" and so on, but not vice versa. All these pronunciations are standard Chinese syllables. We name such a syllable mapping pair as "zhi→ ji" to be a mono-directional fuzzy syllable pair (*FSP*). For simplification, we define the Fuzzy Syllable Set of a syllable *X* as a set of syllables that may be pronounced as *X* in this accent. For example, for Hong Kong Mandarin, the fuzzy set of "ji" is {"zhi", "ji"}. That is to say when a Hong Kong person says "ji", he/she may mean either "zhi" or "ji". For unification, any definite syllable's FSS can be defined as a set of itself.

By this definition, we can modify the SSNS algorithm for the accent Chinese speech recognition. By changing "searching in child nodes for s_l " to "searching in child nodes for any syllable in s_l 's fuzzy syllable set" in Steps 2.1.1 and 2.1.2, the SSNS algorithm will be capable of handling regional accent speech recognition. Of course the prediction function should also be based on the fuzzy syllable pairs.

4. EXPERIMENTS

We have designed and done three kinds of experiments. The first one is to test the word decoding for exact syllable strings. 600 sentences are chosen as the testing word strings. All these sentences are translated into unique syllable strings and taken as the input to the SSNS algorithm one by one. The accuracy is calculated by counting how many output words match the original sentence (word string). For this kind of testing, the word accuracy of syllable-to-word conversion is 99.3%.

The second one is to test the word decoding for a list of syllable string candidates. For the same database as in the first experiment, we make some disturbance to each of them. For each sentence, we randomly change 10% words into different ones (maybe with different length). Including the original one, totally a list of 20 disturbed sentences is turned into syllable strings as the input for testing. In this test, the word accuracy is 96.7%.

The third is to test the accent robustness. Unlike the second test, the syllable string of each sentence is disturbed by some accent syllables into one syllable string. The word accuracy is 99.1%.

5. SUMMARY

In this paper, a syllable-synchronous network search (SSNS) algorithm is proposed originally to offer a solution to the Chinese word segmentation. According to the experimental results, we can draw the following conclusions.

- The SSNS algorithm together with the word search tree (WST) and the N-gram based language model can easily solve the homonym problem in Chinese syllable-to-word conversion. It is useful to the word decoding in the Chinese CSR as well as the character-to-word conversion in the Chinese optical character recognition, because a Chinese word with definite semantic meaning has a unique Chinese syllable string as well as a unique Chinese character string. It automatically segments the syllable string or character string into words.
- The SSNS algorithm offers a solution to the accent Chinese CSR by defining fuzzy syllable sets for a specific regional accent. The solving of accent Chinese speech problem is placed in the language processing stage instead of in the acoustic processing stage.
- The SSNS algorithm tries to solve the homonym, word segmentation and accent problems in the language processing stage, we can think that it diverts the complexity from acoustic stage to language stage. Obviously the Chinese syllable (about 160ms long) is a bigger unit than the speech frame (often 32ms long spaced every 16ms), so roughly it changes the multiplicative complexities into additive complexities.
- The SSNS algorithm reduces the redundancy in the acoustic processing stage caused by the severe homonym problem. Using the SSNS algorithm in the language processing stage, we just need perform syllable-based search in the acoustic processing stage of a CSR system.

6. REFERENCES

[1] Lee C.-H. and Rabiner L. R. "A Frame-Synchronous Network Search Algorithm for Connected Word Recognition," *IEEE Trans. on ASSP*, Nov. 1989, 37(11): 1649-1658.

[2] 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)

[3] Nadas A. "On Turing's Formula for Word Probabilities, " *IEEE Trans. on ASSP*, Vol. ASSP-33, No. 6, 1985

[4] Viterbi A.J. "Error Bounds for Convolutional Codes and An Asymptotically Optimum Decoding Algorithm," *IEEE Trans. on IT-13(2)*, Apr., 1967

[5] Zheng F., Mou X.-L., Xu M.-X., *et al*. "The Implementation of a Speech-to-Text Editor," *5th National Conference on Man-Machine Speech Communication (NCMMSC-98)*, 280-285, 1998 (In Chinese)