Large Vocabulary Speaker Independent Isolated Word Recognition for Embedded Systems

Sergey Astrov, Bernt Andrassy

Corporate Technology
Siemens AG, Munich, Germany
{sergey.astrov, bernt.andrassy}@siemens.com

Abstract

In this paper the implementation of a word-stem based tree search for large vocabulary speaker independent isolated word recognition for embedded systems is presented. Two fast search algorithms combine the effectiveness of the tree structure for large vocabularies and the fast Viterbi search within the regular structures of word-stems. The algorithms are proved to be very effective for workstation and embedded platform realizations. In order to decrease the processing power the word-stem based tree search with frame dropping approach is used. The recognition speed was increased by a factor of 5 without frame dropping and by a factor of 10 with frame dropping in comparison to linear Viterbi search for isolated word recognition task with a vocabulary of 20102 words. Thus, the large vocabulary isolated word recognition becomes possible for embedded systems.

1. Introduction

In the last years several speech recognizers for embedded systems were developed. These recognizers must have a good recognition quality in a noisy environment (for example, street noise, car motor noise). The processor resources and available memory is limited because of the intention to reduce the costs of the recognition platform and reduce the battery power consumption. Currently, realistic system resources for speaker independent speech recognition in embedded devices are about 10-100 MIPS of processing power and several megabytes of memory.

In [1] it was shown that these resources are sufficient for a robust speech recognizer with small vocabulary sizes (about 30-100 words). In this paper the implementation of medium and large vocabulary isolated word recognizer on a low-resource embedded platform is explored.

In a small vocabulary recognition task the most system resource consuming part is the computation of Gaussian log likelihoods. A successful solution to this problem is the subspace distribution clustering hidden Markov models (SDCHMMs) [1].

In the case of a large vocabulary recognition the linear Viterbi search consumes most of the resources. In order to reduce the processing power for large vocabularies the tree search and transducers approaches can be used [2][3]. They proved to be effective solutions for large vocabulary recognition task.

In [4] the word-stem based search was introduced. The approach leads to a very simple and regular algorithm, which can easily be implemented in hardware. In this paper two new modifications of the word-stem based tree search algorithm are considered.

These algorithms are furthermore combined with a so called frame dropping. Based on a voice activity detection, parts of the speech signal that are classified as non-speech are not sent to the recognizer. This further reduces the computational complexity.

The outline of the paper is as follows. Section 2 describes the frame dropping technology. In Section 3 the aspects of the implementation in embedded devices are covered. In Section 4 the experiments are described and the results are shown. The conclusions are made in Section 5.

2. Frame Dropping

Speech signals consist of speech parts and parts during which the speaker is silent. A method to further reduce the computational complexity of the search algorithm is to only send speech parts to the recognizer and drop the non-speech parts. The recognizer processes every 15 ms a frame of 32 ms of the speech signal. A voice activity detection (VAD) classifies each frame into speech or non-speech.

The VAD employed here consists of a multilayer perceptron neural network with error backpropagation during training. The neural network has three layers: input, hidden and output layer. As input 12 cepstral coefficients plus one energy value are taken. The current frame as well as the three past frames and the three future frames are considered leading to 91 input values altogether. As output the net has one node which was trained to represent the non-speech probability of the current frame. A threshold of 0.5 was chosen for this output node to classify into speech and non-speech.

Furthermore a hangbefore of 2 frames and a hangover of 7 frames were applied. Like that seven frames of the signal are sent to the recognizer after the VAD detects the beginning of a non-speech section. The two frames before the VAD detects the beginning of a speech section are likewise sent to the recognizer. Thus a clipping of unvoiced speech parts at the beginning and end of an utterance should be avoided.

This configuration leads to no degradation of the recognition performance in the experiments shown in this paper.

3. Fast Word-Stem Base Tree Search Algorithms

The search problem in the speech recognition is a very computationally complex problem which requires a lot of processing power. The Viterbi algorithm increases the recognition speed. In combination with the pruning approach where only the ”active” states are considered within the search processes, the small vocabulary isolated word recognition becomes possible for embedded devices. The tree search with its processing of equal

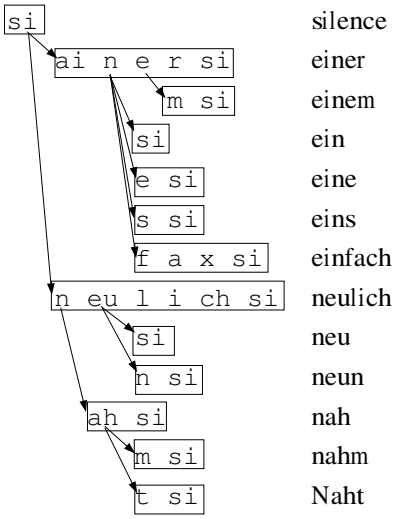

Figure 1: *Word-stem based tree structure for algorithm-1*

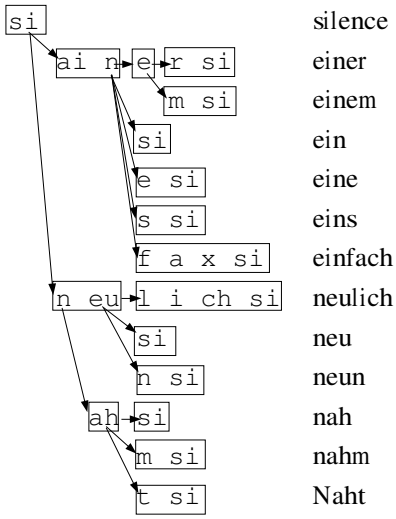

Figure 2: *Word-stem based tree structure for algorithm-2*

word parts of different words only once per frame decreases the required processing power for large vocabulary recognition tasks [2][3].

The implementation of the search algorithms for the large vocabulary isolated word recognition task meets the following problems. For the commonly used tree search a number of complex memory operations is required, as the tree is built up by a complex structure of linked lists. The large address space leads to a huge number of memory access operations and address computations. In embedded systems the cache memory is very small and the cache use may be not very effective (high Cache-Miss-Rate) because of the unregularity of the structure. In [5] the minimized graph search only leads to a modest increase in decoding speed, as the RISC hardware which uses a deeply pipelined ALU cannot function at high efficiency because of the unregularities in minimized graph structure.

In order to accelerate the tree search, the simplier *word-stem based structures* were proposed [4]. Linear word-stem units were used instead of a tree structure that is fully branched after each state. The following two type of units were used: word-stems and word-endings. Word-stems are the first M states of a word, where M is constant. Several words could have a common word-stem which is processed only once per frame. The word-endings are processed for each word independently.

In this paper the word-endings and word-stems are considered as equal units of any length and treated by one algorithm. A word-stem is a set of sequentially placed states, this regular linear structure is easy to process, the Viterbi iteration for a word-stem is computationally effective. The processing data within a word-stem is placed compactly in the memory that is why the cache is used very effectively.

Two fast tree search algorithms were developed. These algorithms code the tree with word-stems in two different ways. The first algorithm aims at small and medium vocabulary speech recognition tasks. The second algorithm is more effective for large vocabulary recognition. In the following the algorithms are described in detail.

The word-stem model for the first algorithm is shown in Figure 1. The word-stems (shown as rectangles) are built on a state level, in Figure 1 they are shown on a phoneme level for a better representation and understanding. This structure has the following features: the end of a word-stem is the end of a word, the branching transitions are possible from any position of a word-stem, the destination state of the transition is always the first state of another word-stem.

In the second algorithm the same tree structure is coded in a different way (see Figure 2). The branching is only possible at the end of a word-stem, the end of a word-stem is not always the end of a word.

Algorithm-1 has less word-stems than algorithm-2, that is why the memory consumption is lower. The total number of word-stems is equal to the number of words plus one (the initial silence state is treated as an extra word-stem).

Algorithm-1 requires an index array of word-stems which stores information about whether the word-stem is active or not active. Even if (almost) all of the word-stems are not active the index array is processed completely, this is the disadvantage of this algorithm. In the case of the very large vocabularies this loop takes the major part of the recognition time. In order to avoid this problem algorithm-2 with its more complex structure is used.

In algorithm-2 a dynamic management of the word-stems is used. The Viterbi iteration is performed only for active word-stems, the size of the active word-stems index array is dynamically changed. The less active word-stems there are, the less processing power is consumed by the algorithm. That is why for large vocabularies algorithm-2 is more computationally effective as algorithm-1.

4. Experiments

In this section the recognition and computation performances of the search algorithm are shown. The processing power of the word-stem based search algorithms and the baseline algorithm are compared. The baseline recognizer is used for research purposes, it has a linear search algorithm.

The algorithms were tested on an isolated word recognition task. The utterances of German city names were recorded over the fixed telephone network and have the sampling frequency of

Table 1: *Memory requirements for the search for different vocabulary sizes*

algorithm	memory requirements			
	495	1500	20k	77k
alg-1	41 KB	120 KB	1.6 MB	4.9 MB
alg-2	53 KB	157 KB	2.1 MB	6.8 MB

8 kHz. Vocabulary sizes of 495, 1500, 20102 and 76784 words were investigated.

The results of the experiments were obtained on a Pentium III 850 MHz workstation running Linux with a test set of 1811 utterances (about 100 minutes of speech with pauses before and after speech parts) in order to explore the recognition accuracy and to compare the performance with the baseline recognizer. The additional tests on an ARM platform were made in order to estimate the required system resources for embedded systems. The processing power for the tree generation is not considered in the experiments as the tree generation was made offline.

The workstation realization always uses 32-bit integers. The ARM realization uses compact coding of parameters in such a way that a 32-bit word contains two or three variables in order to reduce the memory usage and to place the arrays compactly and use the cache effectively.

In Table 1 the memory requirements for the search on the ARM platform are shown for different vocabulary sizes: 495, 1500, 20102 (*20k*) and 76784 (*77k*). As can be observed, the memory consumption is small, a medium vocabulary (500-1500 words) isolated words recognizer could thus be implemented in modern mobile phones and a large vocabulary isolated word recognition could fit into embedded devices (for example, PDAs or car navigation systems).

4.1. Recognition Performance

In this section the word error rate (WER) using N-best search is reported. The search algorithm proposes only N best hypotheses. This type of the recognition could be used in speech controlled navigation systems, e.g. where user may select the result from the proposed list via touch-screen or keyboard.

The two presented algorithms show the same recognition performance as the baseline system, because they realize the same Viterbi algorithm with the same HMM models and pruning algorithm. As said above the frame dropping also led to no decrease in recognition performance.

In Table 2 the recognition results are shown. The N-best recognition rate is shown for $N = 1$, $N = 5$ and $N = 20$. In the columns the WER is shown for different vocabulary sizes. The recognition time includes the reading of feature vectors from files, the Gaussian log likelihoods computation and the Viterbi search. The word-stem based tree, which is generated off-line, is read from the file.

As shown in Table 2 the WER increases with increasing vocabulary sizes. In the experiments the HMM mixtures set for embedded systems with only 1200 Gaussians is used. Large vocabulary recognition tasks (*20k* and *77k*) would require more precise modeling with higher number of Gaussian mixtures.

4.2. Performance Measure for Workstations

In this set of experiments the performance of the search algorithm on a workstation is estimated. The algorithm realization reads the tree file and the feature vectors files of the test set of 1811 utterances. Then the algorithm computes log likelihoods and performs the tree search for each utterance. This test was made with and without frame dropping.

Table 2: *WER for different vocabulary sizes*

N-best	WER [%]			
	495	1500	20k	77k
n=1	13.3	18.6	42.3	59.2
n=5	4.5	6.8	19.5	32.0
n=20	3.5	4.5	9.8	18.2

The execution time per utterance was measured using Linux "time" shell command (user CPU time). These measurements were repeated 5 times and the mean value was used, the results are shown in Table 3. The baseline algorithm was designed for 16-bit word indices, that is why the recognition time for the baseline algorithm could not be measured for 77k vocabulary.

Algorithm-1 and algorithm-2 are in all tests faster than the baseline recognition algorithm, algorithm-2 is always faster than algorithm-1. The frame dropping increases the recognition speed (see lines *baseline-FD*, *alg-1-FD* and *alg-2-FD*). For the baseline algorithm with a 495-word vocabulary the frame dropping increases the recognition time because of the yet unoptimized frame dropping algorithm realization. As shown in Table 3 the frame dropping in general accelerates the search procedure by 2—2.5 times.

Table 3: *Recognition time per utterance for different vocabulary sizes*

algorithm	recognition time per utterance [s]			
	495	1500	20k	77k
baseline	0.12	0.26	2.50	—
baseline-FD	0.15	0.20	1.02	—
alg-1	0.09	0.21	1.89	5.96
alg-1-FD	0.04	0.09	0.68	2.24
alg-2	0.07	0.09	0.47	1.04
alg-2-FD	0.03	0.05	0.21	0.46

Assuming that the typical speech part in the utterances is about 1 s long, the recognition time per utterance can be roughly interpreted as the real-time factor. Algorithm-2 without frame dropping has the recognition time 1.04 s per utterance representing a real-time recognition. Algorithm-2 with frame dropping is 2 times faster than real-time recognition.

4.3. Performance Measure for Embedded Systems

The algorithm performance is explored on an ARM RISC microcontroller which is widely used in embedded systems. The measurements were made according to [1] employing the ARM Instruction Simulator (ARMulator) supplied with the ARM Developer Suite v1.2.

The search algorithms were tested on three ARM processors: ARM9TDMI, ARM920T, ARM940T according to [1], where the same emulation configurations were used. The reference for the maximum performance is an ARM9TDMI uncached core with the ideal zero-wait states 32-bit memory. Two more realistic measurements are made on two configurations: ARM920T core (16KB data and 16KB instruction caches) and ARM940T (4KB data and 4KB instructions caches), for a system configured with the processor clock rate of 100 MHz, a bus clock rate of 50 MHz, and a 32-bit memory with 100 ns non-sequential and 20 ns sequential access time.

Table 4: *Number of core clocks for the recognition task [$\cdot 10^6$]*

vocabulary size	without frame dropping			with frame dropping		
	ARM9TDMI	ARM920T	ARM940T	ARM9TDMI	ARM920T	ARM940T
algorithm - 1						
495	13.5	23.1	27.0	3.7	5.4	6.4
1500	34.7	69.7	74.2	7.8	16.8	17.8
20k	360.4	739.1	731.9	75.0	196.6	194.5
77k	972.8	2392.0	2438.9	337.2	652.2	665.8
algorithm - 2						
495	16.8	24.5	30.8	3.7	5.4	7.0
1500	39.9	71.8	79.6	6.3	14.9	16.6
20k	273.4	600.3	582.5	44.8	98.1	95.5
77k	586.7	1553.9	1548.6	82.2	216.9	217.3

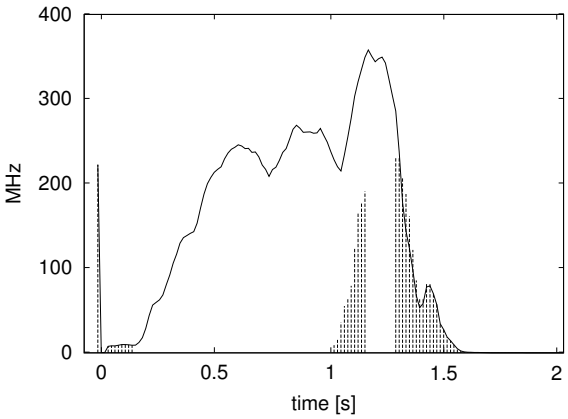

Figure 3: *Dynamically consumed processing power for one utterance (20k vocabulary size)*

In the experiment the program reads the tree file and the precomputed log likelihoods file for the utterance "Tübingen" (255 frames, 3.825 s including pauses before and after the speech part). Then the number of core cycles only for the tree search procedure is counted. The number of million operations per utterance are shown in Table 4.

For the medium vocabulary isolated word recognition (495 and 1500 words) algorithm-1 requires less processing power than algorithm-2. In the case of large vocabulary isolated word recognition tasks algorithm-2 is much faster than algorithm-1. The frame dropping further reduces the recognition time. In general, algorithm-2 with frame dropping has the best results and can be recommended for both medium and large vocabulary isolated word recognition tasks.

The dynamically consumed processing power for ARM9TDMI core is shown in the following experiment. For each frame of the utterance "Tübingen" the required MHz values were computed for isolated word recognition with algorithm-2. The results are shown in Figure 3 for the vocabulary size of 20k words. The dynamically consumed processing power without frame dropping is drawn with the line, the results with frame dropping are drawn with impulses. The processing of the first frame includes the initialization of the search space, that is why the impulses in the beginning of the graphs are observed.

Since the recognizer has a frame buffer, each frame need not be processed in real time. Only 71.5 MHz (273.4 million clocks / 3.825 s) are thus necessary for the recognition of this utterance without frame dropping and only 11.7 MHz (44.8 / 3.825) with frame dropping. As observed, the frame dropping dramatically reduces the required processing power.

5. Conclusions

In this paper new modifications of the word-stem based search for large vocabulary isolated word recognition are developed. The considered algorithms are faster than the baseline search algorithm with linear graph structures. In combination with frame dropping the algorithms allow the realization of large vocabulary isolated word recognition on embedded systems. The approaches are memory and compution ally effective. The search process on isolated word recognition tasks with 1500 words vocabulary sizes requires less than 17 MHz on an ARM processor and 160 KB of memory.

6. References

[1] S. Astrov, J.G. Bauer. S. Stan, "High Performance Speaker and Vocabulary Independent ASR Technology for Mobile Phones", Proc. ICASSP 2003, *accepted paper*, 2003.

[2] S. Kanthak, H. Ney, M. Riley, M. Mohri, "A Comparison of Two LVR Search Optimization Techniques", Proc. ICASSP 2002, pp. 1309–1312, 2002.

[3] H. Dolfing, "A Comparison of Prefix Tree and Finite-State Transducer Search Space Modelings for Large-Vocabulary Speech Recognition", Proc. ICASSP 2002, pp. 1309–1312, 2002.

[4] A. Hauenstein, "Architecture of a 10000 Word Real Time Speech Recognizer", Proc. Eurospeech 1993, pp. 1829–1832, 1993.

[5] S. Deligne et al. "A Robust High Accuracy Speech Recognition System for Moblie Applications", IEEE Trans. Speech and Audio Proc., vol. 10, no. 8, pp. 551–561, 2002.