The Dragon Continuous Speech Recognition System: A Real-Time Implementation

Paul Bamberg, Yen-lu Chow, Laurence Gillick, Robert Roth and Dean Sturtevant
Dragon Systems, Inc.
90 Bridge Street
Newton, MA 02158

Abstract

We present a 1000-word continuous speech recognition (CSR) system that operates in real time on a personal computer (PC). The system, designed for large vocabulary natural language tasks, makes use of phonetic Hidden Markov models (HMM) and incorporates acoustic, phonetic, and linguistic sources of knowledge to achieve high recognition performance. We describe the various components of this system. We also present our strategy for achieving real time recognition on the PC. Using a 486-based PC with a 29K-based add-on board, the recognizer has been timed at 1.1 times real time.

1. Introduction

This paper describes the Dragon continuous speech recognition system that runs in real time on the PC with a 1000-word vocabulary. To achieve the goal of real-time recognition on a personal computer is a process that requires analysis of the computational requirements of the recognition algorithm along several dimensions, and the improving the recognizer's performance along those dimensions.

We first present an overview of the Dragon CSR system architecture, and describe its various components, including signal processing, recognition, rapid match, phonetic modeling, and the application task. We discuss our strategy for achieving real-time continuous speech recognition, and demonstrate how it is actually achieved by developing multiple solutions and applying them in combination.

2. System Description

The architecture of the continuous speech recognition system is shown in Figure 1. The various components of this system are described below.

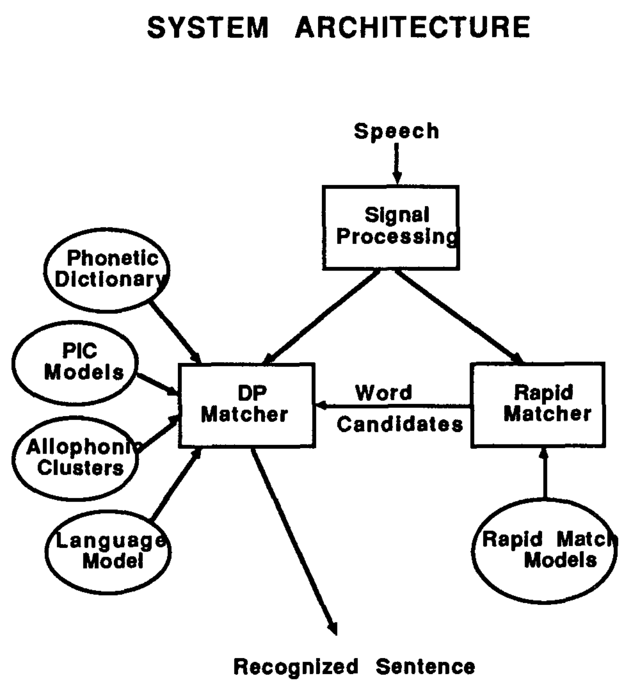

Figure 1: Speech recognition system architecture

2.1 Signal Processing

An TMS32010-based board that plugs into a slot on the AT-bus performs analog-to-digital conversion and digital signal processing of the input speech waveform, and extracts spectral features used in recognition. Input speech is sampled at 12 KHz and lowpass filtered at 6 KHz. Eight spectral parameters are computed every 20 milliseconds, and are used as input to the HMM-based recognizer.

2.2 Recognition

The recognition search to find the most likely sentence hypothesis is based on the time-synchronous decoding algorithm that is used in almost all current CSR systems for this vocabulary size. In this algorithm, partial paths (representing incomplete sentential hypotheses) are extended synchronously using dynamic programming (DP), and all span the same length of the input signal, so that their path cost functions are directly comparable. To reduce recognition search, a beam pruning technique is applied to eliminate all paths that score poorly relative to the best path, and therefore would have very low probability of being the global best hypotheses that spans the entire utterance. We also explored another family of speech decoding algorithms, the stack decoder[1], in our recognizer. It is our conclusion at this time that at least for a task of this complexity, time-synchronous algorithms are considerably more efficient for finding a *single* most likely answer.

2.3 Rapid Matcher

An important component of the recognition search is the Rapid Matcher. In the time-synchronous decoding scheme, the Rapid Matcher helps reduce the search space dramatically by proposing to the HMM DP matcher at any given frame only a relative small number of word candidates that are likely to start at that frame. Only the words on this rapid match list (rather than the entire vocabulary) are considered for seeding a word for DP match. Since the Rapid Matcher is designed to take up considerably less computation than the DP Matcher, the combined rapid match/DP match recognition architecture results in an order of magnitude of savings in computation, with minimal loss in recognition accuracy. The rapid match algorithm is described in detail in [2].

2.4 Training of Acoustic Models

The research goal at Dragon is to build CSR systems for large vocabulary natural language tasks. As such, it is deemed impractical to use whole-word models to model the words in the vocabulary for recognition since in such a system, one must have training tokens (in different acoustic contexts) for every word in the vocabulary. Our solution then is to make extensive use of phonetic modeling for recognition.

In general, the goal of acoustic modeling is to assure that when a string of the acoustic units, whatever they may be, are strung together according to the transcription of an utterance to generate a sequence of spectra, it would fairly accurately represent the actual sequence of speech spectra for that utterance. Towards this goal, we have chosen as the fundamental unit to be trained the "phoneme-in-context" (PIC), proposed in [3]. In the present implementation, a PIC is taken as completely specified by a phoneme accompanied by a preceding phoneme (or silence), a succeeding phoneme (or silence), stress level, and a duration code that indicates the degree of prepausal lengthening. To restrict the proliferation of PICs, syllable boundaries, even word boundaries, are currently ignored.

During training, tokens are phonemically labeled by a semi-automatic procedure using hidden Markov models in which each phoneme is modeled as a sequence of one to six states. A model for a phoneme in a specific context is constructed by interpolating models involving the desired context and acoustically similar contexts.

As each word in the vocabulary is spelled in terms of PICs, each PIC in turn is spelled in terms of *allophonic acoustic segments*, or *clusters*. An acoustic cluster consists of a mean vector and a variance vector. The construction of these clusters is done in a semi-supervised manner. Currently, the total number of acoustic clusters required to construct all PICs is only slightly more than 2000. As a result, the entire set of PICs can be adapted to a new user on the basis of a couple of thousand words of speech data.

With this approach to acoustic modeling, we are able to model words reasonably well acoustically while maintaining to a large extent the desirable property of task-independence. By using different phonetic dictionaries (that make up words for each task), we have constructed models for a 30,000-word isolated word recognizer as well as for four different continuous speech tasks. Details of Dragon's acoustic modeling process can be found in [4].

2.5 Task Description

The Dragon application task consists of recognizing mammography reports. All the training and test material for this task have been extracted from a database of 1.3 million words of mammography text. This text corpus forms part of a 38.2 million word database of radiology text. Much of this text represents actual transcriptions of spoken reports.

All of the test material described here is performed with an 842-word subvocabulary. Punctuation marks, digits, and letters of the alphabet were explicitly excluded. This vocabulary covers about 75% of the full mammography database, and 92% of the database without the excluded words. 6000 sentences (or sentence fragments) containing only these vocabulary words were extracted from the mammography database. Half of these sentences was used for training, and the other half was set aside as test.

2.6 Recognition Performance

Using the system described above, we have obtained preliminary continuous speech recognition results for a 842-word mammography report task, a subset of a full radiology report task. A partial bigram language model was constructed from 40M words of radiology reports, 1M of which was specific to mammography. The bigram language model consisted of unigrams together with common bigrams and uncommon bigrams of common words. The perplexity of this task as measured on a set of 3000 sentences is 66. The result was measured on a single speaker, using 1000 test utterances totaling 8571 words. The total number of word errors was 293 (3.4% word error rate), with 205 substitutions, 62 insertions, and 26 deletions. The sentence error rate was 19.5%. The average number of words returned from the Rapid Matcher (per frame) was 48.

A sample of the test sentences and associated recognition errors made are shown below.

1. These too have increased very slightly
=>
These to have increased very slightly

2. There are no masses demonstrated on today's examination
=>
There are no mass is demonstrated on today's examination

3. The patient returns for additional views for further evaluation
=>
The patient returns for additional view is for further evaluation

We will be evaluating the system on several speakers. In addition, we are working on improving recognition performance, and we have very specific ideas about how that can be done.

3. Real-time Implementation

Our strategy in developing a prototype real-time continuous speech recognition system on the PC is to use a multitude of approaches to solve the computational problem. Since one of our primary concerns is software portability, extensive rewrites in assembly code is kept at a minimum. Instead, we kept almost all of the system written in C, and rely mostly on both algorithm and hardware improvements to achieve real time performance. Software optimizations include the use of a rapid match algorithm to reduce recognition search space, C code optimization, and writing assembly code for a few compute-intensive routines. With hardware, we are relying on the use of both faster machines (e.g., 486-based PC) and more hardware (off-the-shelf boards) serving as compute engines to the PC.

3.1 Algorithms/Software Implementations

Rapid match

The single most important factor in achieving a real time implementation is the use of rapid match to reduce computation during recognition. As described earlier, rapid match is used to compute a relatively short list of likely word candidates that are likely to start at a given frame in the speech input. Thus instead of seeding the entire vocabulary (or close to it), only those words that are returned by the Rapid Matcher are seeded.

Profile and optimize in C

Alternatively, we also invested in profiling the recognition program and getting a report of the amount of time spent in each routine, sorted in decreasing order, so that the first routine on this profiling report is the most time consuming one. Then, if possible, a rewrite of this routine or parts of it with efficiency as the objective is performed. This is done for the top few routines on the list (which usually account for a significant percentage of the total computation). The entire procedure is then repeated.

Assembly language code

Once in a while, as deemed necessary and appropriate, an entire C routine is rewritten in assembly code. Currently, only a few routines have been rewritten this way, which are all routines of the Rapid Matcher.

3.2 Hardware Implementations

A second part of our strategy is to let advances in the technology of manufacturing PCs help in solving the computation problem in continuous speech recognition. Already, we have witnessed an order of magnitude increase in the computation power of a personal computer within the last decade (from AT running at 8 Mhz clock rate to 386 at 33 Mhz). Starting off this decade, the Intel 486-based family of PC's that have just been introduced are a factor 2 faster that its immediate predecessor (the 386-based) machines, given a fixed clock speed of 33 Mhz (see Table 1). This trend will be certain to continue, at least for the foreseeable future. Our recognizer sped up by almost a factor of two just by going from a 386/33 to a 486/33, without any modification to the code (see Table 2). In fact, since the 486 instruction set is downward compatible, the exact same executable code that ran on the 386 also ran on the 486. At this rate, real-time very large vocabulary (> 10,000 words) continuous speech recognition on the PC is within reach not too far in the future.

3.3 Parallel Architecture

We also explored the use of a single (but expandable to multiple) off-the-shelf board (29K-based coprocessor board) serving as compute engine to the PC, and performing the computation in parallel (a coarse grain 2-way parallelism). The board of our choice was an AMD 29000-based board (called the AT-Super made by YARC) that plugs directly onto the AT-bus on the backplane of the PC. The board is quoted at 17 MIPs, although our benchmark in running the recognizer on the board revealed a somewhat lower MIP number (see Table 1). The board also came with some software for development of programs to perform parallel computation.

In analyzing the computation requirements of the various components of our algorithm, it was immediately apparent that a natural way to divide up the algorithm is to have the DP Matcher and the Rapid Matcher run on separate processors, for the following reasons. First, the two components are functionally and logically separable, making parallelization fairly straightforward. Second, it makes sense from the point of the view of the the hardware benchmarks (the two processors give equivalent number of MIPs) as the two recognition components take up within a factor of two relative to the other the number of CPU cycles. Lastly, the communication bandwidth is low (on the order of 5k bytes/sec), so that little overhead is incurred. In the next section, we present results using two alternate ways of mapping the component algorithms to the two processors.

Hardware	Benchmark
386/33 PC	8 MIPs
486/33 PC	15 MIPs
29K Board	12 MIPs

Table 1. Hardware benchmark measured in MIPs.

3.4 Recognition Benchmarks

Table 2 shows the recognition benchmarks (measured in number of times real time) using the various hardware platforms. As can be seen, using a baseline 386 PC, we are at 2.8 X real time. Using a combined 386+29K architecture, and putting the Rapid Matcher on the host and DP Matcher on the 29K (RM/DM) gave us more than a factor of two improvement (to 1.3 X).

Alternatively, going to a faster machine (486-based PC) immediately gave us almost a factor of two relative to running on the 386. However, using the combined 486+29K architecture, though putting us very close to real time (1.1 X), did not provide a significant gain over the 386+29K platform. This is due to the fact that the 29K board, in performing DP match, has become the computational bottleneck. Also, going to the alternative software architecture of performing DP match on the host and rapid match on 29K board (DM/RM) resulted in worse computational performance. This is largely explained by the fact that by performing the rapid match on the 29K, the computational gain that resulted from assembly coding (done for the 386) of some rapid match routines was now lost.

Hardware Arch.	# times real time	
386/33	2.8	
486/33	1.5	
	RM/DM	DM/RM
386+29K	1.3	1.8
486+29K	1.1	1.5

Table 2: Recognition benchmarks with various platforms (# times real time).

3.5 Discussion

Table 3 demonstrates how real-time recognition on the PC was achieved. As noted previously, the use of rapid match to reduce recognition search was the single most important factor in achieving real time. An order of magnitude reduction in computation was realized using this algorithm. Rewriting of C code with runtime efficiency in mind and assembly language coding of some time-critical rapid match routines resulted in factors of 2 and 1.5 speedups, respectively. Finally, making use of more MIPs (either with a 486-based PC or use of a single coprocessor board) gave an additional factor of two to three, depending on the exact hardware platform used. In short, by combining algorithm improvements, software optimizations, and enhanced hardware capabilities, a 3-second long utterance that initially required nearly three minutes to decode (60X real time) now can be decoded in real time.

Method	Speedup
Rapid match	10.0 X
Optimize C code	2.0 X
Assembly	1.5 X
Hardware	2.0 - 3.0 X

Table 3. How real-time recognition was achieved.

4. Conclusion

In summary, we have presented a system for performing 1000-word continuous speech recognition in real time on the personal computer. The system, designed for large vocabulary natural language tasks, is also largely task-independent in that given a new text corpus (used for language modeling) for a new task, we are able to perform recognition on that task within a matter of days.

We also presented our strategies for real-time implementation. Use of advanced algorithms in combination with clever software optimizations, we have reduced computation requirements by a factor of 30, with minimal sacrifice in performance. Using a 386-based PC, the recognizer has been clocked at 2.8 times real time; with a 486-based PC, 1.5 times; and using a 29K-based add-on board, at 1.1 times real time.

REFERENCES

[1] F. Jelinek, "A Fast Sequential Decoding Algorithm Using A Stack", *IBM Journal of Research and Development*, Vol. 13, pp. 675-685, November 1969.

[2] L. Gillick and R. Roth, "A Rapid Match Algorithm for Continuous Speech Recognition", *Proceedings of DARPA Speech and Natural Language Workshop*, June 1990 Hidden Valley, Pennsylvania.

[3] R.M. Schwartz, et al, "Context-Dependent Modeling for Acoustic-Phonetic Recognition of Continuous Speech", *IEEE Int. Conf. Acoust., Speech., Signal Processing*, Tampa, FL, March 1985, pp.1205-1208, Paper 31.3

[4] P. Bamberg and L. Gillick, "Phoneme-in-Context Modeling for Dragon's Continuous Speech Recognizer", *Proceedings of DARPA Speech and Natural Language Workshop*, June 1990 Hidden Valley, Pennsylvania.