INTERNET-ACCESSIBLE SPEECH RECOGNITION TECHNOLOGY[1]

K. Huang and J. Picone

Institute for Signal and Information Processing
Mississippi State University, Mississippi State, MS 39762 USA
{huang,picone}@isip.msstate.edu

ABSTRACT

Speech recognition systems can be viewed as an application of complex pattern recognition and machine learning algorithms. The development of such a system is a time-consuming and infrastructure-intensive task. The Institute for Signal and Information Processing (ISIP) developed one of the first fully-functional public domain speech recognition systems. In this paper, we introduce the major components of this system which include: a digital signal processing front end that generates feature vectors from the speech signal, a hidden Markov Model (HMM) trainer which estimates acoustic model parameters and a hierarchical search decoder which implements an efficient time-synchronous Viterbi beam search.

1. INTRODUCTION

A speech recognition system is an integration of knowledge across several domains such as digital signal processing, natural language processing and machine learning. With the evolution of technology, and the ever- increasing complexity of speech recognition tasks, the development of a state-of-the-art speech recognition system becomes a time-consuming and infrastructure-intensive task.

Since 1994, ISIP has been developing free Internet-accessible software for the speech research community. The development of our speech recognition toolkit began with a prototype system [1] in 1996. The motivation for developing a prototype system was to test implementation of key speech recognition concepts and to understand efficiency issues before we committed to a larger-scale implementation. Based on the prototype system, many applications have been successfully developed including Switchboard (SWB) [2], Call Home [2], Resource Management (RM) [3] and Wall Street Journal (WSJ) [4].

Since 1998, we have focused on the development of a modular and flexible recognition research environment which we refer to as the production system. The toolkit contains many common features found in modern speech to text (STT) systems: a front end that converts the signal to a sequence of feature vectors, an HMM-based acoustic model trainer, and a time-synchronous hierarchical Viterbi decoder.

The key differentiating characteristics of this toolkit include:

- competitive technology with maximum flexibility;
- unrestricted access via the Internet;
- well-documented APIs to facilitate new programming;
- an object-oriented software design.

In this paper, we focus on the three core components of our system: (1) a GUI based tool that lets users implement front ends by drawing block diagrams of signal processing functions; (2) a hierarchical network trainer that simplifies the process of HMM training; and (3) a graph-based generalized hierarchical decoder that supports common features such as lexical trees and cross-unit decoding at all levels of the hierarchy. We also include a brief description of our low-level programming interfaces.

2. A GUI-BASED FRONT END TOOL

An acoustic front end refers to the portion of the recognition system that extracts feature vectors from the speech data. The development of a completely new front end is a software intensive task. The re-implementation of existing algorithms from scratch has slowed down many researchers' efforts. The goal of our front end builder is to provide users an efficient environment for the evaluation of new research ideas. The design of this tool included these requirements:

- rapid prototyping without programming;
- a block diagram approach to describing algorithms;
- a library of standard DSP algorithms and functions;
- an ability to plug in new classes/modules.

The front end builder is designed as a three-level structure so that above features can be achieved. First of all, at the lowest level, a number of core DSP algorithms are implemented as C++ classes, providing high computational efficiency. These algorithms include different types of windows, filters, filter banks, etc. All the algorithms share a common virtual function interface known as an interface contract. Users can plug in their own algorithms into the front end builder by

1.This material is based upon work supported by the National Science Foundation under Grant No. EIA-9809300. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

following the predefined interface specification, without a need to write any new GUI code. Similarly, new algorithm classes can be added to the algorithm library without changing any of the existing code base.

A Java GUI tool was developed to provide users a block diagram approach to designing acoustic front ends. When building a new front end, a user can select from a library of existing algorithms, and can configure all parameters associated with these algorithms from the tool. An example is shown in Figure 1 below. The Java language was used so that the tool can run across a wide range of platforms (including Microsoft Windows), and in order that the tool could be built using an industry-standard look and feel.

The output of the GUI tool is a recipe that schedules the sequences of required signal processing operations. A control program was developed to process the configuration file, pass data through algorithms and generate the final feature vectors. We have successfully built several complex front ends with this tool, including an industry standard front end based on Mel-frequency cepstrum coefficients (MFCCs) [5]. The control program supports multi-pass processing, which allows non-real-time research ideas involving complex normalization and adaptation schemes to be easily implemented.

3. HIEARARCHICAL DECODING

Time-synchronous Viterbi beam search [6] has been the dominant search strategy for continuous speech recognition systems for the past 20 years. Search [1] is a good example of an algorithm that is conceptually simple, but extremely hard to implement in a general way in practice. Worse yet, the slightest inefficiencies in search can result in a system that cannot solve large-scale problems. Most search implementations are restricted to one approach (e.g., time-synchronous Viterbi search) and cannot be separated from the statistical modeling approach (e.g., HMM). Further, the search structure is limited to three levels (e.g., word, phone, and state).

The goals for the design of our generalized decoder were:

- an arbitrary number of independent levels;
- long-span context-dependent models at any level;
- lexical tree expansion at any level;
- posterior symbol probabilities at any level.

We also designed our system to provide alternate algorithm choices for statistical modeling at each state, acoustic modeling via a collection of states, and language modeling via finite state machines. In fact, our decoder is designed to support recognition for applications beyond speech and audio streams. and currently supports multiple search algorithms using a common programming interface.

3.1 Hierarchical Search

A generalized hierarchical search space [7] that must be implemented in a typical speech recognition system is shown in Figure 2. In such a search space, each level can be conceptually considered as the same. Phrases, words or phones are simply symbols at different levels. Each level contains a list of symbols $\boldsymbol{S}_{ij}$ and a list of graphs $\boldsymbol{G}_{ik}$., where, $\boldsymbol{i}$ is the index of the level, $\boldsymbol{j}$ is the index of the symbol and $\boldsymbol{k}$ is the index of the graph. Each graph has at least two dummy vertices, the start vertex and the terminal vertex both indicating the start and the end points of the graph in a search space.

The lower level graph $\boldsymbol{G}_{in}$ (such as $\boldsymbol{G}_{12}$ with **RUN** and **WALK**) is the expansion of the symbol $\boldsymbol{S}_{i-1,n}$ (such as $\boldsymbol{S}_{02}$

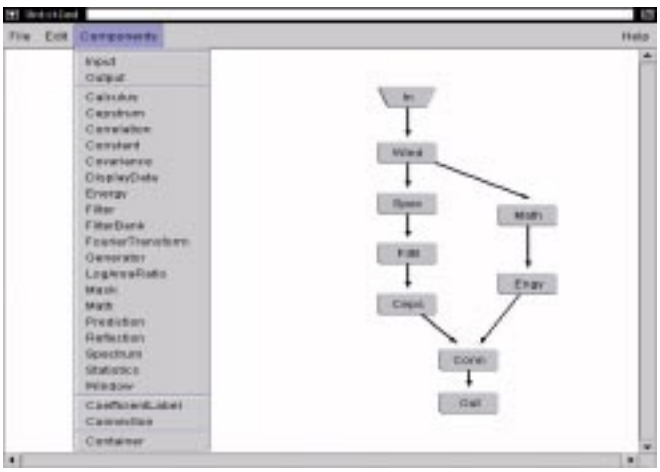

Figure 1: A demonstration of our GUI-based front end configuration tool. Users can create complex front ends using a library of basic algorithm components, and can also easily insert new modules.

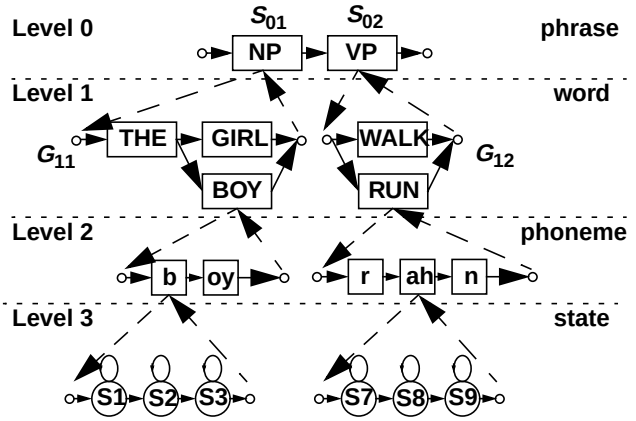

Figure 2: A representation of a generalized hierarchical search space for a recognition system consisting of four levels of knowledge: phrases (sentences), words (dictionary), phonemes (pronunciations) and states (acoustics).

with **VP**) at the level above. This relationship connects all the levels together, realizing the entire search space. There are two exceptions for such a symbol-graph expansion. First, at the highest level, only one graph can exist. This graph is called the *master grammar.* It is a map for the decoder to iterate through the entire search space. Second, at the lowest level, each symbol represents an underlying statistical model. These symbols can not be expanded into a sub-graph. Observations probabilities will be evaluated when the search process reaches these symbols.

The search begins from the start vertex of the *master grammar*, the starting point of the search process. The traces, indicating search paths from one vertex to another, first propagate from the top level to the lowest level, where the observation probabilities of the each frame of data are evaluated. Then the traces propagate forward from the lowest level. Whenever a trace reaches the terminal vertex of a graph, it propagates up. The decoder drives this up and down process for every frame until the end of an utterance. The hypotheses will then be retrieved by traversing backwards through the traces.

3.2 Lexical Tree Expansion

Lexical trees, also referred to as prefix trees, are required so that we can model transitions between words (or any symbol at any level) more accurately. The particular acoustic unit chosen at a boundary can depend on the first sound in the next word. The process of searching for an optimal unit is exponentially complex because we must consider all possible next symbols.

The basic idea of lexical tree expansion is to collapse pronunciation models of different words by sharing the same beginning phonemes. The use of a lexical tree significantly reduces the search space and search effort [8]. Based on the idea of sharing the common prefix, we extend the prefix tree representation to any level in the search hierarchy. If users set the decoder to expand a lexical tree at level *i*, the symbols of level *i* will be expand to their corresponding sub-graphs at level *i+1* and the common prefix of these symbols will be shared, resulting in a tree-structured search.

3.3 Context Dependency and N-gram Decoding

Context-dependent phone models, defined as a model which depends on preceding and following sounds, are generally more accurate than context-independent phone models, since the former can capture coarticulatory effects [6]. In our decoder, the concept of a context-dependent model can be used at any level. If a symbol $\boldsymbol{S}_{ij}$ is context-dependent then the underlying model is determined dynamically via it's neighboring symbols. This generalized implementation imposes no restrictions on the length of context and the number of levels using context. Similarly, N-gram models are extended to N-symbol models. The symbol can be a phrase, a word or a phone, depending on the level at which it is used. The decoder doesn't restrict the order of the N-symbol model. Therefore, users can apply arbitrarily long time-span language or acoustic models to meet the needs of their applications.

4. HMM TRAINING

Our acoustic trainer is a supervised learning machine that estimates the parameters of the acoustic models given the speech data and transcriptions. We support both Baum-Welch and Viterbi re-estimation of model parameters for mixtures of arbitrary statistical models. In addition to standard features of HMM trainers such as state-tying and mixture splitting, additional requirements for our software design included:

- arbitrarily long context-dependent models at any level;
- parallel processing approaches to training;
- network training for enhanced pronunciation modeling.

The HMM trainer we implemented is based on a hierarchical search space described in the previous section. Most STT systems contain three levels of information: the language model (word graph or N-gram), the pronunciation model (lexicon) and the state level (HMM topology for each phone). The motivation to develop a trainer with no restrictions on the number of levels was to further support research into nature language processing, which often involves more than three levels of representation, and uses many diverse knowledge sources.

4.1 Network Training

In most HMM-based systems, a phonetic transcription is required as input to the parameter reestimation process. This transcription is derived at automatically and iterated upon as the recognition system improves. Silence must also be hypothesized between each word model, since we don't know where in the utterance silence actually occurred. In essence, the trainer decides on the optimal alignment of the hypothesized transcription as part of the training process.

Our network trainer, which is depicted in Figure 3, alleviates the need for a phonetic transcription. Instead, it is capable of hypothesizing all possible expansions of a symbol list, and performing parameter re-estimation across this expanded network. The benefits are twofold: we remove the dependency on an intermediate transcription, and we can inherently train statistical pronunciation models. This data-driven approach often results in better performance.

Many existing systems are limited to simple left to right topologies for their acoustic models, thereby restricting the type of research that can be performed with such a system.

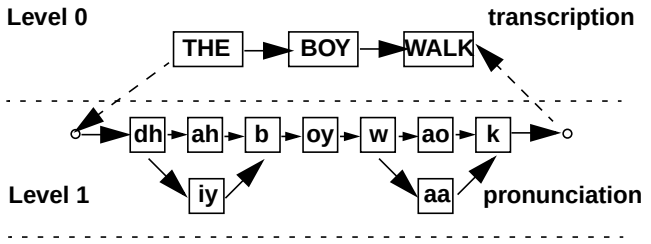

Figure 3: A graph that depicts a representation of multiple pronunciations in network training. Transition probabilities between phones (Level 1) are automatically reestimated using Baum-Welch training.

Our trainer imposes no constraints on the topology of any model in the hierarchical structure. In the training process, models at the top of the hierarchy are dynamically expanded to its constituent models at the lower level, forming a single network. The re-estimation algorithm (e.g., Baum-Welch) operates at all levels and updates parameters related to the transitions and the states simultaneously. Hence, language modeling training is implicit.

5. PROGRAMMING INTERFACES

All the utilities presented above were developed based on an extensive set of foundation classes (IFCs). IFCs are a set of C++ classes organized as libraries in a hierarchical structure. These classes are targeted for the needs of rapid prototyping and lightweight programming without sacrificing the efficiency. Some key features include:

- unicode support for multilingual applications;
- math classes that provide basic linear algebra and efficient matrix manipulations;
- memory management and tracking;
- system and i/o libraries that abstract users from details of the operating systems.

The software environment provides support for users to develop new approaches without rewriting common functions. The software interfaces are carefully designed to be generic and extensible.

6. FUTURE WORK

In this paper, we have presented a brief overview of our long-running project to develop a state-of-the-art public domain speech recognition system. This project began in the fall of 1997, and now supports a customer base of over 150 sites which includes both commercial and government interests. The motivation for the development of such an environment is to provide the community with a toolkit that can accelerate the process of developing new ideas and applications. This toolkit includes many tools to make configuration and operation easy, often without the need to write any code. The system is also unique in that it is built from general purpose math and data structure classes that can be used to pursue other types of research. The complete software toolkit release is available on-line at *http://www.isip.msstate.edu/projects/speech/index.html*.

Our future work will follow two directions: (1) the development of core algorithms to improve recognition accuracy and speed, and (2) integration of natural language processing and dialogue modeling tools. Over the next year, we will release our first real-time dialogue systems application as well as many improvements to the core speech recognition system. If you are interested in tracking our progress, subscribe to the mailing list at the above URL.

7. ACKNOWLEDGEMENTS

There are over 30 students and staff who have made significant contributions to the design and implementation of this software over the past 8 years. We are deeply indebted to them for their hard work and personal sacrifices to make this system a reality. The original design of the system grew out of a series of discussions between I. Alphonso, N. Deshmukh, R. Duncan, A. Ganapathiraju, and J. Hamaker. in 1997. Significant contributions to the front end code have been made by H. Gao.

8. REFERENCES

[1] N. Deshmukh, A. Ganapathiraju and J. Picone, “Hierarchical Search for Large Vocabulary Conversational Speech Recognition,” *IEEE Signal Processing Magazine*, vol. 16, no. 5, pp. 84-107, September 1999.

[2] R. Sundaram, J. Hamaker, and J. Picone, “TWISTER: The ISIP 2001 Conversational Speech Evaluation System,” presented at the Speech Transcription Workshop, Linthicum Heights, Maryland, USA, May 2001.

[3] F. Zheng and J. Picone, “Robust Low Perplexity Voice Interfaces,” *http://www.isip.msstate.edu/publications/reports/index.html*, MITRE Corporation, May 15, 2001.

[4] N. Parihar and J. Picone, “DSR Front End LVCSR Evaluation - Baseline Recognition System Description,” Aurora Working Group, European Telecommunications Standards Institute, July 21, 2001.

[5] S. Davis and P. Mermelstein, “Comparison of Parametric Representations for Monosyllable Word Recognition in Continuously Spoken Sentences,” *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 28, no. 4, pp. 357-366, Aug. 1980.

[6] X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Upper Saddle River, New Jersey, USA, 2001.

[7] B. Jelinek, F. Zheng, N. Parihar, J. Hamaker, and J. Picone, “Generalized Hierarchical Search in the ISIP ASR System,” *Proceedings of the Thirty-Fifth Asilomar Conference on Signals, Systems, and Computers*, vol. 2, pp. 1553-1556, Pacific Grove, California, USA, November 2001.

[8] H. Ney, et al., “Improvements in Beam Search for 10000-word Continuous Speech Recognition,” *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, vol.1, pp. 9-12, San Francisco, California, March 1992.