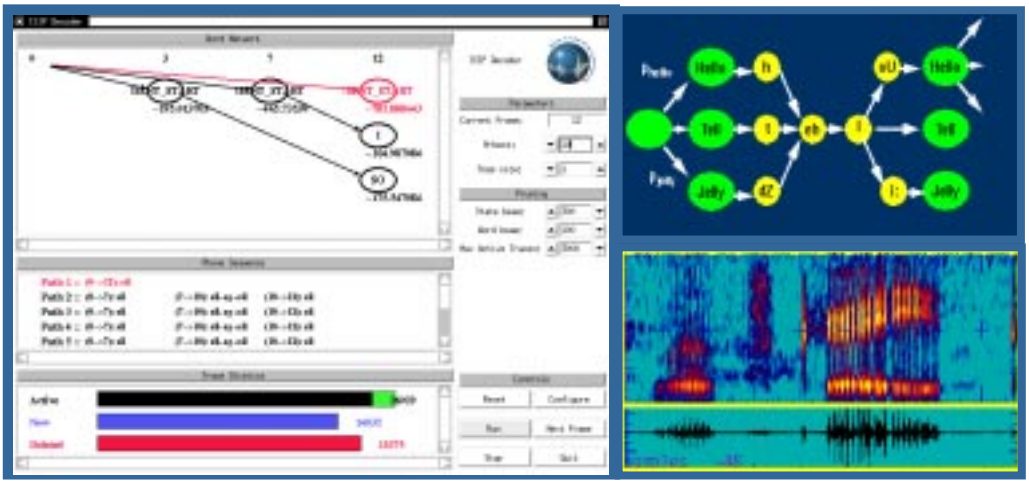
The ISIP Public Domain Decoder for Large Vocabulary Conversational Speech Recognition

Status Report

February 15, 1999



submitted by:

A. Ganapathiraju, N. Deshmukh, J. Zhao, X. Zhang,
Y. Wu, J. Hamaker and J. Picone
Institute for Signal and Information Processing
Department of Electrical and Computer Engineering
Mississippi State University
Box 9571, 413 Simrall, Hardy Road
Mississippi State, Mississippi 39762
Tel: 601-325-3149, Fax: 601-325-3149
primary email contacts: {ganapath, deshmukh, picone}@isip.msstate.edu

EXECUTIVE SUMMARY

Over the past three months, several new components have been added to the ISIP Speech-to-text (STT) system including a Viterbi training module and a front-end capable of generating cepstral features. These features form one complete path to a full-fledged STT system. The decoder has also been reorganized to accommodate the various recognition modes it now supports. The software has been restructured to use more efficient data structures wherever required (such as history nodes to hold all path information). We have also developed a versatile grammar-compiler that compiles any grammar specified in a Bakis-Naur Form (BNF) format into a word graph. The motivation for developing a Viterbi training module instead of the more commonly used Baum-Welch training was that it could be constructed with minimal changes to the core decoder. This module also supports simple model tying and Gaussian mixture component generation and training. The front-end is an independent module that includes features common to most LVCSR systems like cepstral mean subtraction and energy normalization, to name a few.

In the previous release of the decoder, N-gram decoding was achieved by pre-constructing a lexical pronunciation tree that spans all the words likely to appear in the hypotheses, and using copies of this tree with different n-gram histories to hypothesize the next word appropriately weighted with the n-gram (or back-off) likelihood. However, this methodology is very inefficient when handling LVCSR tasks. The decoder has been modified to make this process more efficient. Only one lexical tree is constructed for the lexicon and language model weights are acquired on an as-needed basis. For word graph generation, the decoder software has been fundamentally modified to reflect a hierarchical implementation of paths at the state, phone and word levels (extensible to higher levels such as phrases and sentences), and to allow coexistence of paths with multiple histories i.e. N-best lists. Thus, the same decoding framework can be used to perform n-gram decoding, word graph generation and rescoring. The N-Gram decoding capability of the system has been evaluated using bigrams and cross-word triphones. A word error rate (WER) of 53.2% was obtained, which is comparable to WERs measured by similar research systems at other sites using comparable data and models.

Using all the modules developed thus far, we have built and evaluated a complete system from scratch for a small vocabulary task: Alphadigits. Cepstral coefficients together with delta and acceleration coefficients were used as features. A standard training paradigm developed at WS’97 was used to build this system. Cross-word models with multiple mixture components were trained and evaluated. This system performed close to a system trained using Baum-Welch training: 15.2% WER for ISIP’s Viterbi-based system vs. 13.5% for an HTK Baum-Welch based system). On-line web-based documentation for this system was also introduced. A step-by-step tutorial for training an alphadigit system from scratch is included in this documentation. The complete system, including documentation, is available in the public domain at *http://www.isip.msstate.edu/projects/speech_recognition/*.

We are currently in the process of evaluating the system on a much more complex task: SWITCHBOARD. Our goal for the next three months is to have a system capable of generating rich word graphs using a trigram and cross-word context-dependent triphones. Modules that allow rescoring of word graphs using different language models without any acoustic rescoring will be provided. This should be very useful for sites involved in language modeling research.

TABLE OF CONTENTS

1. ABSTRACT

Following a demonstration of the decoder on a SWITCHBOARD (SWB) [1] word graph rescoring task at the Hub-5 Conversational Speech Recognition (LVCSR) Workshop sponsored by the DoD [2], the STT toolkit project shifted its focus to the enhancement of basic functionality. Several new components, most notably signal processing, training, and grammar compilation, were added to the system. The STT system is now capable of running complete recognition experiments, and ready to be tested on new applications. The system is now capable of training both context-independent and context-dependent models with multiple Gaussian mixture components using the Viterbi algorithm. The system also supports model tying. A new feature extraction module capable of generating cepstral features was also added. A complete system, including web-based documentation and step-by-step tutorials is now available for public use.

2. INTRODUCTION

The progress of Speech to Text (STT) systems on conversational speech has been slow in recent years. Reasons for this include the complexity of the task and the infrastructure required to deal with this complexity. The prohibitively high entry-level costs for research in this area have motivated the Institute for Signal and Information Processing (ISIP) to establish a center of excellence for speech recognition technology that will provide free access to state-of-the-art STT technology. Our primary goal is to leverage state-of-the-art STT technology and harness the Internet as a means to share resources and provide unrestricted global access to the relevant software and data.

Some of our goals include:

- unrestricted access (via ftp and WWW)
- detailed documentation and instructions for operations (e.g. a tutorial)
- state-of-the-art technology and periodic upgrades
- object-oriented software design for extensibility by the user / developer
- on-line technical support

We were very pleased to release our first self-contained STT system on February 1, 1999. This was the culmination of one and a half years of research and engineering, and the pot of gold at the end of a rainbow that formed almost ten years ago.

2.1. Previous Deliverables

The previous version of the ISIP decoder released in November 1998 had the functionality of most state-of-the-art decoders. Almost all the major data structures were handled by the central memory manager of the decoder to minimize the load on the system due to the decoder memory requirements. Improved heuristics for path merging and pruning were developed to increase the efficiency of decoding. The decoder supported the following modes of operation —

- word graph rescoring
- network grammar decoding
- forced alignment
- ngram decoding

The decoder had the ability to dynamically construct lexical trees to predict next-phone hypotheses, and perform accurate and efficient rescoring of word graphs while maintaining relatively low requirements on system memory. The details of this system are described in [3].

2.2. Limitations in Previous Versions

The previous version of the decoder did not have the capability of generating word graphs in N-Gram decoding mode. This is an essential feature for decoders used in LVCSR tasks. Another drawback of the previous design was the fact that the same data structure was used to store history and path information. This creates problems in the computation of acoustic scores of paths in the word graph generation mode. In order to correct this flaw, we needed to design a new data structure that holds history information and is independent of the path information that is propagated at every frame. The N-Gram decoding module's implementation involved the creation of several copies of the lexical tree during decoding. However this is prohibitively expensive in terms of system memory, especially when dealing vocabularies in the order of a few tens of thousands words. The decoding algorithm needed to be modified to tackle this inefficiency.

2.3. Deliverables

As described previously, latest version of the ISIP decoder has undergone numerous design changes as well as enhancements of its functionality. All major data structures are handled by the central memory manager of the decoder to minimize the load on the system due to the decoder memory requirements. The decoder now supports the following modes of operation:

- word graph rescoring (acoustic/language model)
- network grammar decoding
- word graph generation
- forced alignment
- ngram decoding

We have added modules that perform HMM training, feature extraction and grammar compilation. The training module is capable of performing Viterbi estimation of model parameters and can handle model tying and multiple Gaussian mixture components per state. The feature extraction module is capable of generating cepstral features, including delta and acceleration coefficients. Some of its advanced features include cepstral mean normalization and energy normalization. Also included in the current release is a versatile grammar compiler that compiles a grammar specified in the BNF format [5] to a word graph that can be used as input to the decoder. With the current release of the STT system we have also released web-based documentation and tutorials.

3. ENHANCEMENTS TO THE ISIP DECODER

The goal of the ISIP STT toolkit is to make a state-of-the-art speech recognition system available to scientists such that new fundamental research can be efficiently executed. The decoder, which is easily the most complicated part of this system, must be modular and extensible in order to deliver state-of-the-art recognition performance on tasks of widely differing complexities (such as connected digits, studio-quality read speech, and spontaneous telephone conversations) in a transparent fashion.

3.1. The ISIP Decoder Functionality

The core search algorithm used in the ISIP decoder is based on a hierarchical variation of the standard Viterbi-style time-synchronous search paradigm [4]. At each frame of the utterance, the system maintains complete history for each active path at each level in the search hierarchy via special scoring data structures (markers). Each path marker keeps its bearings in the search space hierarchy by indexing the current history (or word graph) node, lexical tree node and the triphone model. It also maintains the path score and a backpointer to its predecessor.

For each instantiation of a triphone model, a state-level path marker is projected from the previous phone-level marker and added to a state-level list of path markers. For each frame, the active states are evaluated only once. The state-level markers are compared and the best marker for each different instance of the state is projected to the next states as governed by the state transition probabilities (Viterbi decoding). The score for each state is stored locally and added to the projected path marker score. A marker exiting the model is added to the phone-level marker list and used to project the next triphone markers. Similarly, phone-level path markers at word endings are promoted to the word level and used to project paths into subsequent words.

Lexical Trees

Compared to past releases of the decoder, the lexical tree organization has undergone a fundamental change in the way word information is stored. The motivation for this redesign was the need to reduce ambiguity in the comparison of word-level path markers. Previously, the word end nodes consisted of a list of words that end at that node. This is however conceptually ambiguous and requires the path markers to carry too much unnecessary information. A decoding algorithm would be much cleaner if each word-end node represented a unique node. Figure 1 depicts an example of such an implementation.

History Information

One of the most important changes in the decoder in this release relates to the problem of word graph generation, and the amount of history information that must be propagated at every stage of the decoding process. Since word graph generation inherently involves tracking multiple histories at the end of each word, a careful design of the structure that carries this information is needed. In the previous release of the decoder, this information was part of the path marker. However, computing the acoustic scores for each arc of the word graph at the end of the decoding process is inaccurate in this case. To remedy this problem and to keep the history and path information separate, we decided on adding a new data-structure to hold the history information. Though this change marginally increased memory usage, it was deemed not significant enough to warrant making the decoder less flexible.

3.2. ISIP Decoder Operating Modes

The ISIP decoder can be used to perform decoding in various ways depending on the application and the requirements of the user. The software is designed such that a single decoding framework can be used to handle many different modalities, and the search process is fairly generalized and

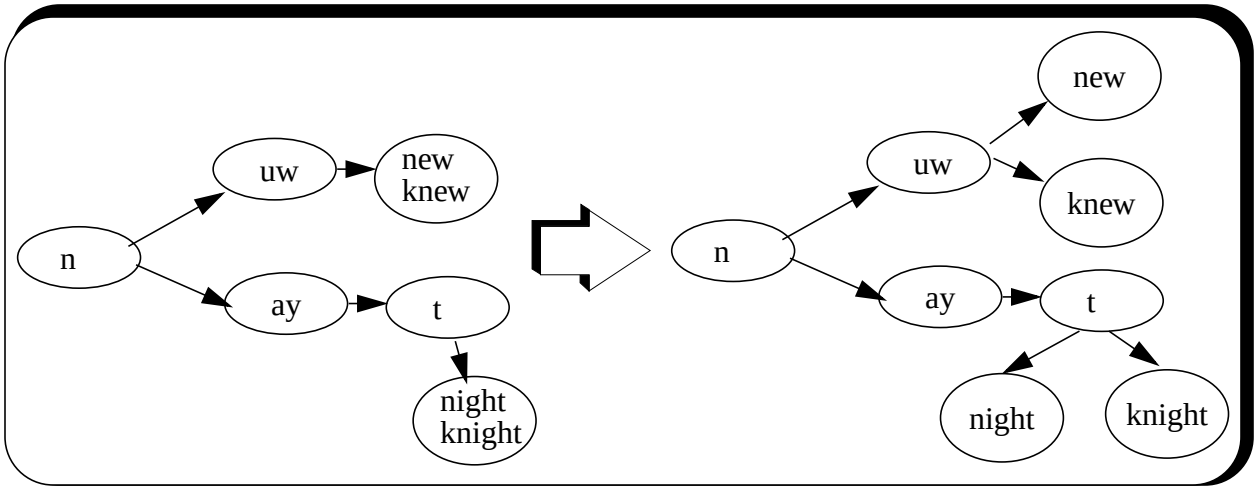


Figure 1. A comparison of the current implementation of the lexical tree definition with that used in the previous release.

transparent (there is little exception handling, and most of the code is common to all these functionalities). In all cases, the decoder expects options to be specified via a parameter file. The main operating modes currently supported by the decoder are briefly described below.

Word Graph Rescoring

In word graph rescoring mode, a file containing a list of word graphs is used as an additional input. Each word graph file consists of a graph derived in a previous recognition pass. The ISIP decoder refines the search space using the word start and stop times embedded in the graph, and rescores the acoustic models to generate the final hypothesis. By applying different acoustic and linguistic constraints during this rescoring pass, the user can improve recognition performance.

Forced Alignment

Forced alignment refers to the process of aligning the speech signal with a given reference transcription at the state, phone and/or word level. Forced alignment is the foundation upon which acoustic model training is implemented. In this mode, the utterance transcription completely defines the linguistic search space. No hypothesis other than that described by the reference transcription is allowed to be generated. The decoder requires a file containing such reference transcriptions as an additional input in this mode.

Network Grammar Decoding

We have built a grammar compiler utility that takes as input a file containing a grammar specified in a BNF format and converts it into a word graph format. This word graph can be used as an input to a word graph rescoring mode to introduce linguistic constraints to the search space. In this way, a network grammar decoding can be provided that is completely consistent with the basic search engine is implemented. The grammar compiler currently exists as a stand-alone utility. It will be integrated into the decoder package very soon.

N-Gram Decoding

This mode uses an N-gram language model (LM) such as a bigram or trigram (with back-off) to define the linguistic search space and predict the next possible word(s) [8]. The decoder requires an input file containing this language model data which it reads into special data structures designed to efficiently compute LM scores. The decoder constructs a lexical tree encompassing all the words in the lexicon. This tree is then used to generate phone hypotheses. The scores for word transitions are computed based on the position in this lexical tree and the current N-gram history. A virtual copy of the tree is created for each N-gram word history.

Word Graph Generation

The word graph generation mode creates an ordered list of multiple hypotheses at each word end, keeping track of the different path histories of that word. At the end of the decoding process, all the paths terminating at a proper end of sentence position in the search space are output in the form of a word graph that consists of a word at each node. The arcs indicate word transitions that constitute different paths. Each arc is associated with an acoustic score for the end word and a linguistic likelihood of the end word given the start word of that arc. During word graph generation, the linguistic search space is typically constrained by a grammar or an N-gram language model.

4. ISIP FRONT-END

The most commonly used front-end in speech recognition systems today is based on a cepstral analysis of the speech signal. The portion of the STT system that derives such features from the speech signal is referred to as the signal processing module, or the front-end. Our initial approach to developing a versatile front-end was to first develop a cepstral front-end capable of generating industry-standard cepstral coefficients, along with coefficients that describe their derivatives (delta) and acceleration (double-delta). Key features of this front-end, such as cepstral mean subtraction, have been implemented. These coefficients has been verified to be interchangeable with those generated from standard commercial packages (this was no small task). The following section describes the algorithms in greater detail.

4.1. Mel Frequency Cepstral Coefficients

A mel is a psychoacoustic unit of measure for the perceived pitch of a tone, rather than the physical frequency. The correlation of the mel to the physical frequency is not linear, as the human auditory system is a nonlinear system. A mapping between the mel scale and real frequencies was empirically determined. The scale is roughly linear below 1000 Hz, and then logarithmically beyond 1000 Hz. This transformation can be described mathematically as:

$$Mel(f) = 2595\log_{10}(1 + f/700) \quad (1)$$

This compression of the frequency scale, when implemented as above, allows a front-end to be constructed using a bank of digital filters, or several other similar approaches operating directly in

the frequency or autocorrelation space. These filters can be evenly spaced along the mel scale, as depicted in Figure 2. The six highest triangular filters are overlaid on the spectrum of a speech signal. The vertical bars below this figure represent the amplitude of the output of these filters.

A homomorphic system is useful for speech processing because it offers a methodology for separating the excitation signal from the vocal tract shape. One space which offers this property is the cepstrum, computed as the inverse discrete Fourier transform (IDFT) of the log energy. This signal is by definition minimum phase, another useful property. Cepstral coefficients are computed by the following equation:

$$c(n) = \frac{1}{N_s}\sum_{k=0}^{N_s} \log\left|S_{avg}(k)\right| e^{j\frac{2\pi}{N_s}kn} \qquad 0 \le n \le N_s - 1 \tag{2}$$

A critical analysis of the cepstral variability across different speakers and channel conditions leads to a more robust acoustic model for automatic speech recognition. The higher order cepstral coefficients are more influenced by algorithmic artifacts of the LPC analysis (the all-pole constraint, for instance). Alternately, the low cepstral coefficients vary primarily due to variations in transmission, speaker characteristics, and vocal efforts. A liftering procedure,

$$w(n) = \begin{cases} 1 + \left(\frac{L}{2}\right)\sin\frac{n\pi}{L} & n = 1, 2, \ldots, L \\ 0 & n \le 0, n > L \end{cases} \tag{3}$$

is used to weight the cepstrum and control the non-information bearing variabilities. For

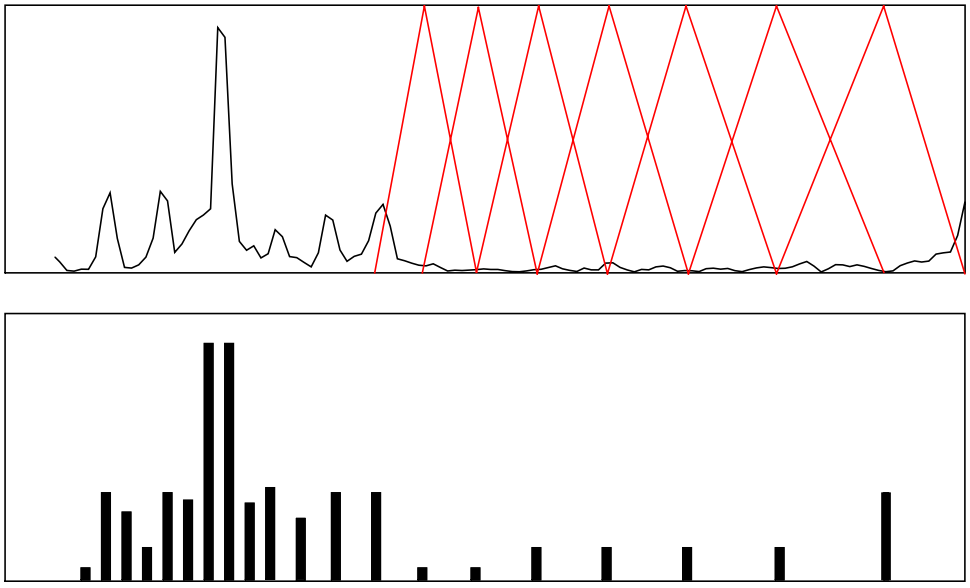


Figure 2. Mel-frequency spaced triangular filters used in our cepstral front-end.

telephone bandwidth speech, typically L is set to 24 [6]. Most state-of-the-art speech recognition systems use a front-end that consists of 12 Fourier transform-derived mel-frequency cepstral coefficients and mean energy as a first order model of the signal.

4.2. Delta and Acceleration Coefficients

Delta and acceleration coefficients have been proven to produce improved recognition results as they compensate for the lack of temporal and transitional information in the normal cepstral features. These coefficients are derived from the normal coefficients via a set of weighted difference equations [6].

$$d_n = \frac{\sum_{w=1}^{dw} w(c_{n+w} - c_{n-w})}{2\sum_{w=1}^{dw} w^2} \tag{4}$$

where d_n is a delta coefficient at frame n, c_{n-w} and c_{n+w} are static parameters for the frames before and after the current frame coefficient c_n, and dw is the delta window size which is typically one. In order to compute the acceleration coefficients, the same formula is used, except that the static coefficients are now replaced by their delta coefficients.

4.3. Advanced Features

Several other features that make a front-end robust to noise and compensate for the artifacts of frame-based computations are also implemented in the ISIP front-end. The front-end gives the user a wide range of windowing functions to choose from, including the standard Hamming and Hanning windows. Standard signal processing features like, pre-emphasis and energy normalization are also supported. Cepstral mean normalization, a technique used for noise removal, and a technique that is the dominant noise removal and channel normalization procedure in the industry, is an integral part of the system. In fact, combination of all features described above will produce signal features that interoperate with other commercially available systems.

5. VITERBI TRAINING MODULE

As mentioned previously, though the Baum-Welch algorithm is the most effective and commonly used parameter estimation procedure used in contemporary speech recognition systems, the Viterbi algorithm is attractive due to its simplicity and ease of implementation. The decoder required very limited changes to accommodate training.

5.1. Theory

In Viterbi training, the mean vectors and covariance matrices for the observation probability densities are estimated by simple averaging [7]. For simplicity, assume that there is only one mixture component per state. Each input observation vector maps to one state in the best state

sequence that maximizes the probability of the data given the model. Suppose there are N_i observation vectors that mapped to ith state while training on several utterances. Then the reestimated mean and covariance for ith state are computed as:

$$\mu_i = \frac{1}{N_i} \sum_{n=1}^{N} y(n), \text{ and} \tag{5}$$

$$C_i = \frac{1}{N_i} \sum_{n=1}^{N} [y(n) - \mu_i][y(n) - \mu_i]^T. \tag{6}$$

When handling multiple mixtures we use a widely used heuristic to update mixture weights. Since the acoustic score at any state in the HMM is computed as a cumulative over all mixtures, there are two alternatives to assign vectors to different mixtures: we map the vector to all mixtures in the state or we map the vector to the mixture that contributes the most to the score at that state. In our training module we choose the later. Once we assign vectors to each component we use Eqs. 5 and 6 to update the means and variances for each mixture.

5.2. Implementation

Training in our STT system is currently organized as set of utilities in which the Viterbi training module forms the core. Organizing the training paradigm in such a distributed fashion has several advantages including restarting experiments at various stages of the training process. This is very attractive while training large systems which typically need vast amounts of disk space and hundreds of hours of compute time.

The various utilities and their functions are listed below:

- init_hmm: initializes monophone models using a global mean and a diagonal covariance
- init_triphones: initializes triphone models based on information regarding the clustering of models
- create_mixtures: allows generation of an arbitrary number of Gaussian mixture components
- hmm_train: the core Viterbi training module

We also expect that users will train models that are initialized using the commonly used commercial speech recognition toolkit, HTK [6]. To support this we have utilities that convert HTK models to formats specific to the ISIP recognition system.

6. EVALUATIONS

Since the last release, the most important additions to the system are the ability to perform N-Gram decoding and training context-dependent models. Both these features have been evaluated on commonly used test data. N-Gram decoding capability was evaluated on SWB and the training module was evaluated on a small vocabulary task, OGI Alphadigits [9].

6.1. N-Gram Decoding Using Bigram Language Models

N-Gram decoding is highly resource intensive compared to word graph rescoring. It typically requires about 5 times as much time as word graph rescoring with a similar scale factor for memory usage. With the intention of expediting evaluations, we choose to test this feature on a subset of the standard SWB evaluation data. The bigram language model used has been used to generate word graphs over the past few workshops at Johns Hopkins University. It contains 22,000 unigrams and 300,000 word pairs with backoff [8].

For this dataset, rescoring of word graphs gives a WER of 50.6%. This is summarized below in Table 1. Note however that the word graphs are bound to give better performance because they were generated using triphone models enhanced by speaker adaptation and vocal tract length normalization.

6.2. Cross-Word Models For Alphadigits

At ISIP, we have used the OGI Alphadigits task to test several systems in the past. This task has a 40 word vocabulary which restricts the number of context-dependent models to a manageable number. However, since the word error rates on this specific database are in the 10% range, this task can be used to calibrate system performance effectively without having to deal with the massive amount of resources needed to train a full fledged SWB system.

In order to evaluate the complete STT system, we extracted features from the raw data and trained context-dependent phone models with 8 Gaussian mixture components per state. These models were then tested using the decoder. The grammar for this task was compiled into a word-graph using the grammar compiler. Thus all components of the STT system were used in building this system. Figure 3 depicts the steps in building this system.

The industry standard cepstral front-end described previously was used as the feature set. A training paradigm similar to what was used in WS’97 was followed to train context-dependent triphone models. The training set consisted of 15,000 utterances averaging about 6 digits and alphabets in length. For the flat-start stage, a tenth of the complete training data was used.

As a point of comparison, a system of comparable complexity was built using HTK. The main

	Bigram	Word Graph
WER	53.2%	50.6%
Subs.	36.1%	35.4%
Dels.	13.6%	9.8%
Ins.	3.4%	5.3%

Table 1: Bigram decoding vs. rescoring of bigram word graphs.

difference being the training algorithm: HTK used Baum-Welch and ISIP used the Viterbi training algorithm. Table 2 summarizes the results obtained using both the systems.

7. SUMMARY

We have delivered a complete STT system capable of extracting features, training acoustic models and decoding test utterances. Though this system does not include all the algorithms common to most state-of-the-art STT systems, it does provide a means to perform research and achieve reasonable performance. The system presently includes a Viterbi training module, a front-end capable of generating cepstral features and a grammar compiler that can handle any grammar specified in the BNF format (a format commonly used to represent small vocabulary tasks). We have modified the ISIP decoder to enhance many of its existing features such as word graph generation, and word graph rescoring.

We have also demonstrated the functionality of the complete system by training an Alphadigit recognition system from scratch. The recognition performance of the ISIP decoder is comparable to systems trained using commercial software (15.6% WER vs. 14.5% WER for HVite): the difference being attributable our use of the Viterbi algorithm (versus Baum-Welch). Preliminary evaluation of the decoder on bigrams for SWB is encouraging and comparable to results reported in the literature. The complete STT toolkit, including the decoder, training module, front-end and grammar-compiler, is freely available at the ISIP web site:

http://www.isip.msstate.edu/projects/speech_recognition

Documentation for the various utilities and a tutorial are also available on-line at the same URL.

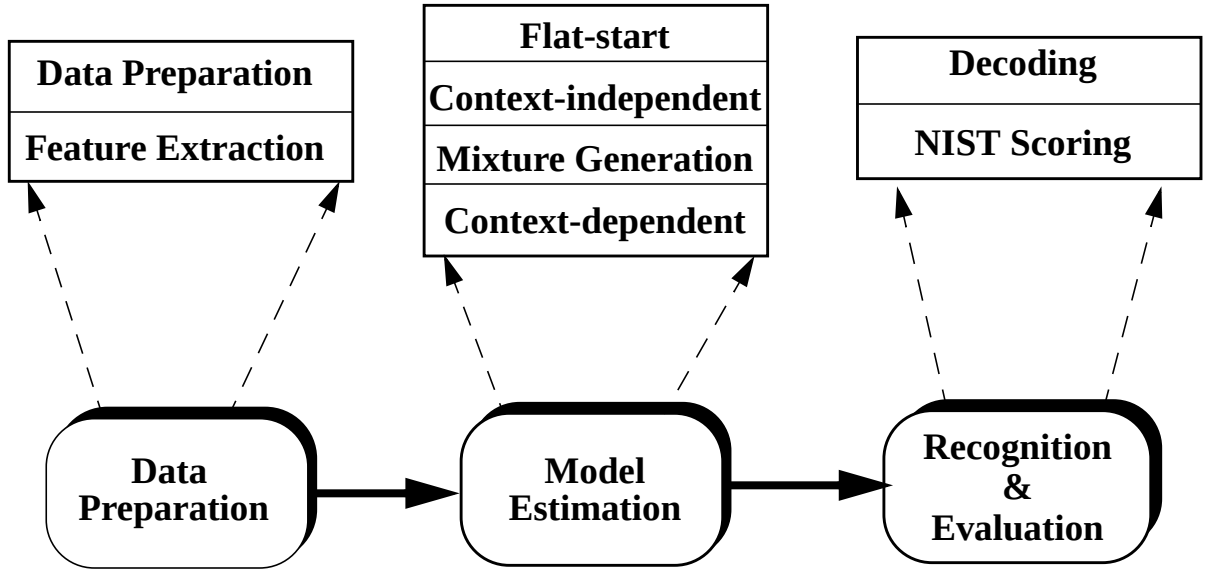


Figure 3. Training and evaluation of context-dependent cross-word phone models

	ISIP	HTK
WER	15.6%	14.5%
Subs.	13.8%	12.1%
Dels.	1.0%	0.3%
Ins.	0.8%	2.0%

Table 2: A comparison of systems trained using HTK and ISIP's STT toolkit.

8. FUTURE WORK

Despite delivering a working system, the variety of algorithms at each point in the system is limited. Our focus in the next phase of this project will be directed towards adding new algorithms and expanding the flexibility of the system.

Decoder

We plan on evaluating the word graph generation capabilities of the decoder under several scenarios. A typical evaluation paradigm to verify the richness of word graphs is to rescore word graphs using a language model that is more complex than the one used for generating the word graph (for example, generating graphs with a bigram, and rescoring with a trigram). This process does not involve acoustic models and is very handy in evaluating new language models. Its counterpart for acoustic modeling is when we evaluate new acoustic models using the same language model as the one used for word graph generation. This procedure is especially useful for research in developing new acoustic models. We plan on adding functionality to the decoder to handle these cases. This will make the system much more attractive for researchers.

Feature Extraction

We currently are in the process of developing a comprehensive front-end module for the speech recognition system. This module will implement several standard front-ends, and be built on top of a general framework for doing DSP research (known as the ISIP DSP foundation classes). The framework for this module has being carefully designed to ensure simple integration with the speech recognition system. Adding new algorithms is fairly easy and often simply a matter of changing a parameter file. We plan on extending the capability of this module to include several commonly used features — perceptual linear prediction, filter bank amplitudes, and RASTA filtering [10]. The current implementation also expects input data to be in a 16-bit linear format. However this is quite restrictive. We plan on supporting μ–law data and data stored in the NIST SPHERE format [11] as well.

Training

Training involves creation of the acoustic and language models to be used with the decoder. Most acoustic parameter estimation algorithms are based on the expectation-maximization (EM) procedure [12]. Baum-Welch or the forward-backward algorithm is one such algorithm. Though it

is more involved computationally than the Viterbi algorithm, it has several advantages over the Viterbi algorithm, including insensitivity to initial model settings. We plan on incorporating this algorithm into the training module. Another very important feature of most state-of-the-art recognizers is a decision-tree clustering module used to cluster context-dependent models in order to compensate for lack of sufficient training data for the vast number of models [13]. This feature will be added to the recognizer in the near future.

9. REFERENCES

[1] J. Godfrey, E. Holliman, and J. McDaniel, "SWITCHBOARD: Telephone Speech Corpus for Research and Development," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, San Francisco, California, USA, pp. 517-520, March 1992.

[2] N. Deshmukh, A. Ganapathiraju, J. Hamaker, and J. Picone, "An Efficient Public Domain LVCSR Decoder," *Proceedings of the Hub-5 Conversational Speech Recognition (LVCSR) Workshop*, Linthicum Heights, Maryland, USA, September 1998.

[3] N. Deshmukh, A. Ganapathiraju, J. Hamaker, and J. Picone, "An Internet-Based Public Domain Speech-to-Text Toolkit," *Quarterly Status Report for the Department of Defense*, Institute for Signal and Information Processing, Mississippi State University, November 1998.

[4] S. J. Young, N. H. Russell and J. H. S. Thornton, "Token Passing: A Simple Conceptual Model for Connected Speech Recognition Systems," *Cambridge University Engineering Department Technical Report CUED/F-INFENG/TR.38*, Cambridge University, UK, 1989.

[5] J. H. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley Publishing Company, Reading, Massachusetts, USA, 1979.

[6] P. Woodland, et.al., *HTK Version 1.5: User, Reference and Programmer Manuals*, Cambridge University Engineering Department & Entropic Research Labs. Inc., 1995.

[7] J. R. Deller, J. G. Proakis, and J. H. L. Hansen, *Discrete Time Processing of Speech Signals*, Macmillan Publishing Company, New York, New York, USA, 1993..

[8] R. Rosenfeld, "A Maximum Entropy Approach to Adaptive Statistical Language Modeling," *Computer, Speech and Language*, vol. 10, pp. 187-228, 1996.

[9] R. Cole et. al., "Alphadigit Corpus," *http://www.cse.ogi.edu/CSLU/corpora/alphadigit*, Center for Spoken Language Understanding, Oregon Graduate Institute, 1997.

[10] H. Hermansky and N. Morgan, "RASTA Processing of Speech," *IEEE Transactions on Speech and Audio Processing*, pp. 578-589, 1994.

[11] D. Pallett et. al., "Speech File Manipulation Software: SPHERE," *http://www.itl.nist.gov/*

div894/894.01/software.htm, NIST Spoken Natural Language Processing Group, February 1999.

[12] A. P. Dempster, N. M. Laird and D. B. Rubin, "Maximum Likelihood from Incomplete Data via EM Algorithm," *Journal of the Royal Statistical Society*, pp. 1-38, January 1997.

[13] J. J. Odell, V. Valtchev, P. C. Woodland and S. J. Young, "A One-Pass Decoder Design for Large Vocabulary Recognition", *Proceedings of the DARPA Human Language Technology Workshop*, pp. 405-410, March 1995.