Large-Vocabulary Speech Recognition Algorithms

By making the advances necessary to implement next-generation speech recognition applications, researchers could develop systems within a decade that match human performance levels.

Mukund Padmanabhan
Michael Picheny
IBM T.J. Watson Research Center

Providing the computer with a natural interface, including the ability to understand human speech, has been a research goal for almost 40 years. Speech recognition research started with an attempt to decode isolated words from a small vocabulary. As time progressed, the research community began working on large-vocabulary and continuous-speech tasks. Practical versions of such systems have become moderately usable and commercially successful only in the past few years, however. Even now, these commercial applications either restrict the vocabulary to a few thousand words, in the case of banking or airline reservation systems, or require high-bandwidth, high-feedback situations such as dictation, which requires modifying the user's speech to minimize recognition errors.

Early attempts at speech recognition tried to apply expert knowledge about speech production and perception processes, but researchers found that such knowledge was inadequate for capturing the complexities of continuous speech. To date, statistical modeling techniques trained from hundreds of hours of speech have provided most speech recognition advancements. Speech researchers have combined these modeling techniques with the massive increase in available computing power over the past several years to explore complex models with hundreds of thousands of parameters.

Multisite speech recognition research cooperation and competition, supported through government agencies such as DARPA, have also fueled advancements in this field. In addition to participating in government-sponsored competitions, industrial labs, universities, and other companies have fostered rapid advances in speech recognition technology by sharing data and algorithms (http://www.nist.gov/speech/publications/tw00). A by-product of these cooperative efforts has been that most successful systems share roughly the same architecture and algorithms because each site immediately copies other sites' successful algorithms.

To enable next-generation applications such as speech recognition over cellular phones, transcription of call center interactions, and recognition of broadcast news, researchers continue to work on the advanced speech recognition (ASR) problem. An understanding of today's ASR systems architecture provides a basis for exploring the recent advances motivated by next-generation applications.

BASIC ARCHITECTURE

The process of speech recognition starts with a sampled speech signal. This signal has a good deal of redundancy because the physical constraints on the articulators that produce speech—the glottis, tongue, lips, and so on—prevent them from moving quickly. Consequently, the ASR system can compress information by extracting a sequence of *acoustic feature vectors* from the signal.

Typically, the system extracts a single multidimensional feature vector every 10 ms that consists of 39 parameters. Researchers refer to these feature vectors, which contain information about the local

0018-9162/02/$17.00 © 2002 IEEE

frequency content in the speech signal, as *acoustic observations* because they represent the quantities the ASR system actually observes. The system seeks to infer the spoken word sequence that could have produced the observed acoustic sequence.

To simplify the design, researchers assume that the ASR system knows the speaker's vocabulary. This approach restricts the search for possible word sequences to words listed in the *lexicon,* which lists the vocabulary and provides *phonemes*—a set of basic units, usually individual speech sounds—for the pronunciation of each word.

Commercial lexicons typically include tens of thousands of words, however, and the length of the word sequence uttered by the speaker is unknown. Let us assume that the length of the word sequence is N. If V represents the size of the lexicon, the ASR system can hypothesize V^N possible word sequences. Language constraints alone dictate that these word sequences are not equally likely to occur. For example, the word sequence "give me a call" is much more likely than "give call a me." Further, the acoustic feature vectors extracted from the speech signal provide significant clues about the phoneme that produced them.

The sequence of phonemes that corresponds to the acoustic observations implies the word sequence that could have produced the sequence of sounds. Consequently, the acoustic observations provide an important source of information that can help further narrow the space of possible word sequences.

The ASR system uses this information to assign a probability that the observed acoustic feature vectors were produced when the speaker uttered a particular word sequence. Essentially, the system efficiently computes these probabilities and outputs the most probable sequence as the *decoded hypothesis*.

All of today's most successful speech recognition systems use a generative probabilistic model that encapsulates this sequence of steps. As Equation 1 shows, in this model the recognizer seeks to find the word sequence $\hat{w}_1^N$ that maximizes the word sequence's probability, given some observed acoustic sequence y_1^T. This approach applies Bayes' law and ignores the denominator term to maximize the product of two terms: the probability of the acoustic observations given the word sequence and the probability of the word sequence itself.

$$\hat{w}_1^N = \arg\max_{w_1^N} p\left(w_1^N \middle| y_1^T\right) \equiv \arg\max_{w_1^N} p\left(y_1^T \middle| w_1^N\right) p\left(w_1^N\right) \quad (1)$$

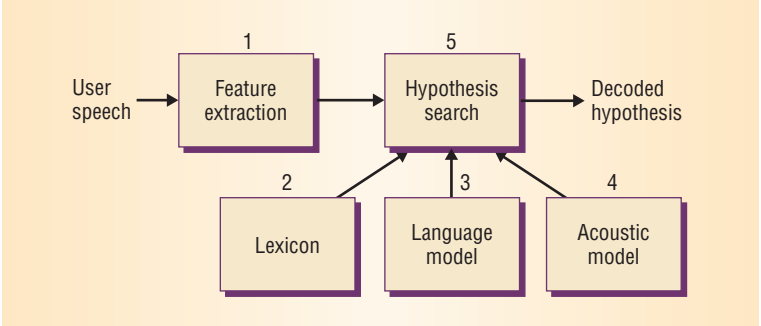

Figure 1. Speech recognition system block diagram. The lexicon, language model, and acoustic model components construct a hypothesis for interpreting a speech sample.

Figure 1 shows the process described by Equation 1 as a block diagram. In Block 1, the search extracts multidimensional features from the sampled speech signal. In Block 5, the search hypothesizes a probable word sequence. Several components drive this step:

- the lexicon in Block 2 defines the possible words that the search can hypothesize, representing each word as a linear sequence of phonemes;
- the language model in Block 3 models the linguistic structure but does not contain any knowledge about the relationship between the feature vectors and the words, $(p(w_1^N|y_1^T))$; and
- the acoustic model in Block 4 models the relationship between the feature vectors and the phonemes, $(p(y_1^T|w_1^N))$.

Getting the best performance for feature extraction and hypothesis searches requires customizing the ASR system.

Feature vectors

When working with feature vectors, the most commonly used schemes extract a multidimensional vector from the sampled speech signal at a uniform frame rate, typically every 10 ms. These feature vectors provide clues about the phonemes that produced them. Consequently, the procedure for extracting them is modeled on the workings of the human auditory system.

The human auditory system simulates a constant-Q-filter bank, in which the sensitivity to the energy in each channel follows a logarithmic relationship. In a constant Q-filter bank, the ratio of the center frequencies of adjacent filters is constant, and the filter bandwidth is proportional to the center frequencies. Most feature-extraction schemes mimic these steps, with some variations in modeling perceptual aspects, such as mel-frequency versus perceptual linear prediction cepstral parameters.[1,2]

Because the temporal variation of the speech signal's local spectral content also contains information that helps infer spoken phonemes, the system often computes and appends the temporal derivatives and second derivatives of these features to form the final feature vector.

Table 1. Typical lexicon, with the word *the* having two pronunciations.

Word	Phonetic representation
The	DH AH
The	DH IY
Cat	K AE T
Pig	P IH G
Two	T UW

HYPOTHESIS SEARCH

Three basic components comprise the hypothesis search: a lexicon, a language model, and an acoustic model.

Lexicon

The typical lexicon shown in Table 1 lists each word's possible pronunciations, constructed from phonemes, of which English uses approximately 50. An individual word can have multiple pronunciations, however, which complicates recognition tasks. The system chooses the lexicon on a task-dependent basis, trading off vocabulary size with word coverage. Although a search can easily find phonetic representations for commonly used words in various sources, task-dependent jargon often requires writing out pronunciations by hand.

Language model

The search for the most likely word sequence in Equation 1 requires the computation of two terms, $p(y_1^T|w_1^N)$ and $p(w_1^N)$. The second of these computations is called the *language model*. Its function is to assign a probability to a sequence of words w_1^N.

The simplest way to determine such a probability would be to compute the relative frequencies of different word sequences. However, the number of different sequences grows exponentially with the length of the sequence, making this approach infeasible.

A typical approximation assumes that the probability of the current word depends on the previous two words only, so that the computation can approximate the probability of the word sequence as:

$$p\left(w_1^N\right) \approx p(w_1)p\left(w_2|w_1\right)\prod_{i=3}^{i=N} p\left(w_i|w_{i-1},w_{i-2}\right) \quad (2)$$

The computation can estimate $p(w_i|w_{i-1}, w_{i-2})$ by counting the relative frequencies of word trigrams, or triplets:

$$p\left(w_i|w_{i-1},w_{i-2}\right) \approx N\left(w_i,w_{i-1},w_{i-2}\right)/ N\left(w_{i-1},w_{i-2}\right) \quad (3)$$

where N refers to the associated event's relative frequency. Typically, training such a language model requires using hundreds of millions of words to estimate $p(w_i|w_{i-1}, w_{i-2})$. Even then, many trigrams do not occur in the training text, so the computation must smooth the probability estimates to avoid zeros in the probability assignment.[3]

Acoustic models

An acoustic model computes the probability of feature vector sequences under the assumption that a particular word sequence produced the vectors.

Given speech's inherently stochastic nature, speakers usually do not utter a word the same way twice. The variation in a word's or phoneme's pronunciation manifests itself in two ways: duration and spectral content, also known as acoustic observations. Further, phonemes in the surrounding context can cause variations in a particular phoneme's spectral content, a phenomenon called *coarticulation*.

Hidden Markov models. A *hidden Markov model* offers a natural choice for modeling speech's stochastic aspects. HMMs function as probabilistic finite state machines: The model consists of a set of states, and its topology specifies the allowed transitions between them. At every time frame, an HMM makes a probabilistic transition from one state to another and emits a feature vector with each transition.

Figure 2 shows an HMM for a phoneme. A set of *state transition probabilities*—$p1$, $p2$, and $p3$—governs the possible transitions between states. They specify the probability of going from one state at time t to another state at time $t + 1$. The feature vectors emitted while making a particular transition represent the spectral characteristics of the speech at that point, which vary corresponding to different pronunciations of the phoneme. A *probability distribution* or *probability density function* models this variation. The functions—$p(y|1)$, $p(y|2)$, and $p(y|3)$—could be different for different transitions. Typically, these distributions are modeled as parametric distributions—a mixture of multidimensional gaussians, for example.

The HMM shown in Figure 2 consists of three states. The phoneme's pronunciation corresponds to starting from the first state and making a sequence of transitions to eventually arrive at the third state. The duration of the phoneme equals the number of time frames required to complete the transition sequence. The three transition probabilities implicitly specify a probability distribution that governs this duration. If any of these transitions exhibits high self-loop probabilities, the

model spends more time in the same state, consequently taking longer to go from the first to the third state. The probability density functions associated with the three transitions govern the sequence of output feature vectors.

A fundamental operation is the computation of the likelihood that an HMM produces a given sequence of acoustic feature vectors. For example, assume that the system extracted T feature vectors from speech corresponding to the pronunciation of a single phoneme, and that the system seeks to infer which phoneme from a set of 50 was spoken. The procedure for inferring the phoneme assumes that the ith phoneme was spoken and finds the likelihood that the HMM for this phoneme produced the observed feature vectors. The system then hypothesizes that the spoken phoneme model is the one with the highest likelihood of matching the observed sequence of feature vectors.

If we know the sequence of HMM states, we can easily compute the probability of a sequence of feature vectors. In this case, the system computes the likelihood of the tth feature vector, y_t, using the probability density function for the HMM state at time t. The likelihood of the complete set of T feature vectors is the product of all these individual likelihoods. However, because we generally do not know the actual sequence of transitions, the likelihood computation process sums all possible state sequences. Given that all HMM dependencies are local, we can derive efficient formulas for performing these calculations recursively.[3]

Parameter estimation. The state transition probabilities and the gaussian means and variances that model the feature vectors' probability density functions parameterize the HMM's different states. Before using an HMM to compute the likelihood values of feature vector sequences, we must train the HMMs to estimate the model's parameters. This process assumes the availability of a large amount of training data, which consists of the spoken word sequence and the feature vectors extracted from the speech signal.

Researchers commonly use the maximum likelihood estimation process training paradigm for this task. Given that we know the correct word sequence corresponding to the feature vector sequence, the ML estimation process tries to choose the HMM parameters that maximize the training feature vectors' likelihood, computed using the HMM for the correct word sequence. If y_1^T represents the stream of T acoustic observations, and w_1^N represents the correct word sequence, the ML estimate for the parameter θ is

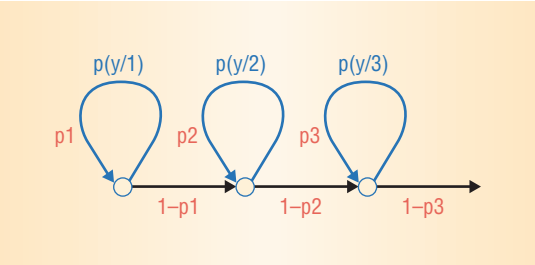

Figure 2. Hidden Markov model for a phoneme. State transition probabilities p1, p2, and p3 govern the possible transitions between states.

$$\hat{\theta}_{ML} = \arg\max_{\theta} \log\left[p_\theta\left(y_1^T \middle| w_1^N\right)\right] \quad (4)$$

The system begins the training process by constructing an HMM for the correct word sequence. First, it constructs the HMMs for each word by concatenating the HMMs for the phonemes that comprise the word. Subsequently, it concatenates the word HMMs to form the HMM for the complete utterance. For example, the HMM for the utterance "We were" would be the concatenation of the HMMs for the four phonemes "W IY W ER."

The training process assumes that the system can generate the acoustic observations y_1^T by traversing the HMM from its initial state to its final state in T time frames. However, because the system cannot trace the actual state sequence, the ML estimation process assumes that this state sequence is hidden and averages all possible state sequence values.

By using s_t to denote the hidden state at time t, then making various assumptions, the system can express the maximization of Equation 4 in terms of the HMM's hidden states, as follows:

$$\arg\max_{\theta} \sum_{t=1}^{T} \sum_{s_t} p_\theta\left(s_t \middle| y_1^T\right) \log\left[p_{\hat{\theta}}\left(y_t \middle| s_t\right)\right] \quad (5)$$

The system uses an iterative process to solve Equation 5, with each iteration involving an expectation step and a maximization step.[3] The first step involves the computation of $p_\theta(s_t|y_1^T)$, which is the posterior probability—or count of a state—conditioned on all the acoustic observations. The system uses the current HMM parameter estimates and the Forward-Backward algorithm[3] to perform this computation. The second step involves choosing the parameter $\hat{\theta}$ to maximize Equation 5. When the probability density functions are gaussians, the computation can derive closed-form expressions for this step.

Coarticulation. So far, we have assumed that the fundamental acoustic units are phonemes and that the system uses HMMs to model the duration and acoustic variation associated with the phonemes' pronunciation. However, in some cases, the phonemes in the surrounding context affect a particular phoneme's acoustic variation. This coarticulation phenomenon is particularly prevalent in spontaneous speech, which the speaker does not enunciate carefully. The system models coarticulation by assuming that the density of the obser-

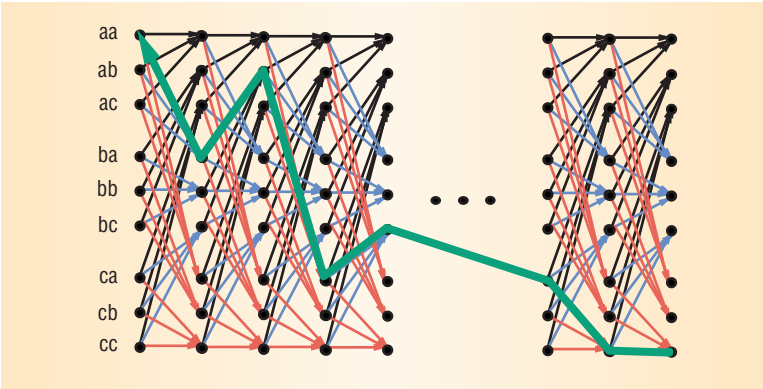

Figure 3. Hypothesis search for a trigram language model, with states indicated on the left for $w_{n-1}w_{n-2}$ pairs. The diagram indicates the w_n transitions for words a, b, and c with black, blue, and red, respectively, and shows a typical traceback in green.

vations depends on both the specific phoneme and the surrounding phonemes.

However, modeling every phoneme in every possible context generates a prohibitively large number of densities that need modeling. For example, if the phoneme alphabet consists of 50 phonemes, and the system models every phoneme in the context of its immediately surrounding neighbors, it would need to model 125,000 densities. Consequently, a commonly adopted approach clusters the surrounding phonemes into a few equivalence classes, thus reducing the densities that require modeling.[3] Typical large-vocabulary speech recognition systems contain thousands of context-dependent densities that capture different coarticulatory behaviors.

Search process

The hypothesis search seeks the word sequence with the highest likelihood, given the model's input features and parameters.[3] Because the number of word sequences increases exponentially with the word sequence's length, the search might seem at first to be an intractable problem for anything other than short utterances from a small vocabulary. However, because the model has only local probabilistic dependencies, the system can incrementally search through the hypotheses in a left-to-right fashion, discarding most candidates with no loss in optimality.[3]

Figure 3 shows a *recognition trellis*, a diagram commonly used for recognition searches. In this simple example, the vocabulary contains three words: a, b, and c. Each small circle represents a state of the language model. In this diagram, the trigram language model has nine states, representing each possible combination of the vocabulary's two possible preceding words. For each language model state, the search could output any of the three words in the vocabulary. The search activates a different set of language model states depending on which word it outputs. For example, if the search outputs the word "a," it will output the states labeled aa, ab, and ac at the next interval. Thus, at a given point, the search can determine the best path leading into a particular language model state.

For example, three paths lead into state aa: the sequences aaa, baa, and caa. The language model's local dependency lets the search discard the two lowest-scoring paths and use a back pointer to track the path with the highest score. The decision about the overall best path must wait until the search reaches the end of the sentence. The system then recursively follows the set of back pointers from the highest scoring path at the sentence's end to determine the overall best path through the sentence.

Although the number of states in the language model can theoretically grow as the square of the vocabulary, many trigrams never actually occur in the training data. The smoothing operation backs off to bigram and unigram estimators, substantially reducing trellis size. To speed up the recursive process, the system can conduct a *beam search*, which makes additional approximations such as retaining only hypotheses that fall within a threshold of the maximum score in any time frame.

BENCHMARKING ASR SYSTEM PERFORMANCE

The *word error rate* quantifies ASR system performance by expressing, as a percentage, the ratio of the number of word errors to the number of words in the reference transcription. Depending on the task, ASR system word error rates can vary by a couple of orders of magnitude. Several factors affect the word error rate:

- *Vocabulary size*. Small vocabulary tasks, such as digits, have a lower word error rate.
- *Language model perplexity*. This characteristic indicates the language's branching power, with low-perplexity tasks generally having a lower word error rate.
- *Background noise*. The cleaner the background in which the test speech is recorded, the lower the word error rate.
- *Speech spontaneity*. The more spontaneous the speech, the higher the word error rate.
- *Sampling rate*. The higher the speech signal sampling rate, the better the performance.
- *Amount of training data available*. More training data means better performance.

We used the word error rate results on a voicemail transcription task to report these results. This

task uses ASR systems to transcribe messages that consist of spontaneous telephone speech—sampled at 8 KHz—with a fairly large vocabulary. Consequently, this effort represents one of the more challenging ASR research tasks.

Although we quantified the relative improvements achieved using this task, we can apply these quite general methods to any speech recognition task. The test data consisted of 105 voicemail messages comprising 52 minutes of speech. The training data consists of 4,700 voicemail messages comprising 53 hours of speech. Using this task as a baseline, we achieved a speaker-independent word error rate of 40.5 percent.

RECENT ASR SYSTEMS ADVANCES

In the past few years, several advances, including significant enhancements to accuracy, have improved the performance of individual ASR system components. The broad classifications for these improvements include

- novel methods of extracting acoustic observations from the speech signal,
- alternatives to Maximum Likelihood estimation of the HMM parameters,
- postprocessing methods for hypothesizing a better sequence of words,
- adaptation of the acoustic models based on limited amounts of test data from the speakers, and
- methods for combining the output of several ASR systems.

These improvements have been incorporated into most commercial and laboratory systems.

Feature extraction

Developers often augment the feature vector by using first and second derivatives to encode it with temporal trajectory information. If the extracted cepstra are d-dimensional, they can generate an acoustic feature that has 3xd dimensions. A more sophisticated approach uses a linear discriminant transformation to incorporate such information. This approach concatenates the d-dimensional cepstra from several adjacent frames, typically nine, to form a 9xd-dimension feature vector. It reduces the feature vector's dimensionality by computing a linear projection that finds the directions that maximally separate the phonetic classes. The system then uses a mixture of gaussians to model the probability density function of each HMM state's projected feature vectors. Because of computational considerations, most ASR systems assume that these gaussians have diagonal covariance matrices, which in effect assumes that the projected feature vectors have independent dimensions.

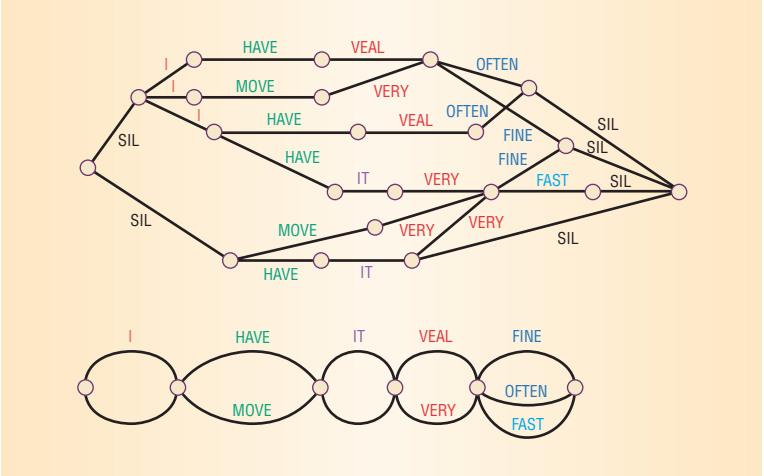

Figure 4. Converting a word lattice to a confusion network by merging different paths in the graph. The resulting chain's components represent parallel word sequences.

More specifically, we use the Maximum Likelihood Discriminant projection.[4] This method simultaneously maximizes the log likelihood of the data in the projected space and the separation between the class means in the projected space, while minimizing the correlation between the projected feature vectors' dimensions. Using this projection reduces the baseline system's word error rate from 40.5 percent to 39.1 percent.

Hypothesis search

The most commonly used decoding paradigm for speech recognition is the maximum-a-posteriori (MAP) rule. In an alternative procedure for scoring the hypothesis search, w_1^N represents the decoded word sequence and w'^N_1 represents the correct word sequence. This procedure defines a loss function $l(w_1^N, w'^N_1)$ that quantifies the difference between the two word sequences, then further defines the decoding procedure's objective as minimizing the average expected loss.

This procedure can be written as

$$w_1^{N*} = \underset{w_1'^N}{\operatorname{argmin}} \sum_{w_1^N} l\left(w_1^N, w_1'^N\right) p\left(w_1^N \middle| y_1^T\right) \qquad (6)$$

If $l(w_1^N, w'^N_1)$ is a delta function that represents the sentence error rate, then Equation 6 reduces to the commonly used MAP decoding rule. Hence, the MAP decoding rule essentially minimizes the sentence error rate in a hypothesis search. However, given that most speech recognition applications focus on the *word* error rate, it makes more sense

Speaker-independent systems provide generic acoustic models that match the feature vectors for any speaker reasonably well.

to define $l(w_1^N, w'^N_1)$ as the word error rate between the two word sequences.

We can use this to obtain a "consensus" hypothesis that minimizes the loss.[5] As Figure 4 shows, this procedure uses a MAP decoder to compute a word lattice graph, then merges different paths in the graph to convert the word lattice into a chainlike structure. The chain's components represent parallel word sequences. Minimizing the loss is equivalent to picking the most probable word in each component, and concatenating these words represents the consensus hypothesis. Using this consensus search technique reduces the word error rate from 39.1 to 38.0 percent.

Acoustic model adaptation

Generally, speaker-dependent ASR systems—which developers train to recognize speech from a single speaker—provide the best performance. Building such a model requires a large amount of training data from the selected speaker, which can be difficult to obtain. ASR systems can, however, collect a small amount of speech data—a few minutes—which they then use to either adapt a speaker-independent system's parameters to the speaker or to adapt the feature-extraction process for the speaker.

Adapting the model also requires transcription of the speech data. If the correct transcription is available, the procedure is a *supervised adaptation*. This kind of procedure usually uses adaptation methods such as MAP estimation[6] and maximum likelihood linear regression (MLLR).[7]

If the correct transcription is unavailable, a speaker-independent system is used to provide an (errorful) transcription; the same procedure is applied as above, but is referred to as *unsupervised adaptation*. Surprisingly, even with baseline error rates as high as 80 percent, unsupervised adaptation can improve recognition performance if the system carefully constrains the adaptation parameters.

Vocal tract length normalization. In ASR systems, the acoustic model's probability density functions model the feature vectors that derive from statistical variations in speech. Given that the system estimates these probability density functions from the speech of several different speakers, the functions model not only the inherent statistical variation within a phoneme, but also the variations in the vocal tract characteristics of different speakers.

The vocal tract length normalization (VTLN)[8] adaptation method attempts to remove the interspeaker component of the variation in the models, and is applied during the feature extraction stage. The idea is to extract acoustic observations in such a way that the feature vectors for a particular phonetic class look similar for different speakers. This is equivalent to mapping the speech from a given speaker to a *canonical space* that characterizes a *canonical speaker*. The acoustic model parameters are then estimated from the canonical feature vectors.

Developers based the VTLN method on the observation that the variance in the pitch and format ranges is the most significant difference between speakers. Consequently, scaling the power spectrum so that the speaker's format frequencies take on a target value—the canonical speaker's format frequency value—could eliminate interspeaker variability in the acoustic feature vectors. Using VTLN to normalize the feature vectors reduces the word error rate from 38.0 to 35.0 percent.

Linear transform adaptation. Speaker-independent systems provide generic acoustic models that match the feature vectors for any speaker reasonably well. Although it is a major strength, this characteristic also limits the system: A speaker-dependent system built from a large amount of data generated by the same speaker provides a better match for the speaker's feature vectors. Consequently, the speaker adaptation method uses a limited amount of adaptation data from the test speaker, and it changes the speaker-independent models to bring them closer to speaker-dependent models.

A commonly used method[7,9] focuses only on the parameters of the gaussian probability density functions that make up the acoustic model, ignoring the transition probabilities. This method assumes that the system obtains the adapted probability density functions by applying an affine transformation to the means and covariances of the speaker-independent probability density functions.

This method uses the adaptation data to estimate the parameters of these transformations. The ASR system then uses the adapted probability density functions to transcribe speech from that speaker. Using these linear transform adaptation methods reduces the word error rate from 35.0 to 34.3 percent.

System combination

Although most ASR systems exhibit a similar average performance, there is a significant difference in their outputs. For example, an ASR system that uses different feature representations—such as mel-frequency and perceptual liner predictive para-

meters—could produce hypotheses that differ by as much as 20 percent.

An alternative method aligns multiple ASR outputs to form a word transition network for the utterance.[10] The WTN is formed through an iterative process that aligns the first two ASR system hypotheses. Subsequently, the process aligns the next ASR hypothesis to the WTN, and the next, and so on. Finally, a voting or scoring module searches the composite WTN to select the best scoring word sequence. This method treats the output of multiple ASR systems as independent knowledge sources, then combines these outputs to produce a composite output with a lower word error rate than an individual ASR output.

We applied this technique to combine the hypotheses of four systems, trained with different feature sets and model sizes, that exhibited word error rates of 39.0 percent, 34.3 percent, 38.7 percent, and 34.4 percent. The technique reduced the final word error rate to 32.7 percent, an overall relative reduction of 19 percent from the initial 40.5 percent error rate.

COMPARISONS WITH HUMAN PERFORMANCE

Recent advances have improved the word error rate from 40.5 to 32.7 percent on a voicemail-transcription task. To place this task's performance in context, consider the performance level of transcriptions of several other speech recognition tasks, as listed in Table 2.[11]

Just the fact that we can now transcribe voicemail and conversational speech with 30 to 40 percent error rates and dictation with 7 percent error rates represents major progress in the speech recognition field. However, despite this progress, recent studies suggest that machine performance still lags behind human performance by at least an order of magnitude across a wide variety of tasks, ranging from high-bandwidth digit recognition to large-vocabulary telephony speech.[11] Further, human recognition performance across signal-to-noise ratios is highly robust—less than 20 percent degradation from quiet to ratios of 10 dB—while even highly compensated machine performance typically degrades by at least a factor of two over such a range.

Despite steady ASR system progress, the word error rate for spontaneous speech sources remains relatively high, and the systems remain sensitive to background noise. Humans, in contrast, often perform an order of magnitude better, and they also show a more robust resistance to background noise.

Table 2. Machine versus human word error rates across a variety of tasks.[11]

Task	Machine performance	Human performance
Connected digits	0.72%	0.009%
Letters	5.0%	1.60%
Transactional speech	3.6%	0.10%
Dictation	7.2%	0.90%
Conversational telephone speech	43.0%	5.00%

Although it may appear that we have far to go before ASR systems can match human performance, if researchers maintain the current rate of yearly progress in reducing word error rates, that objective should be within reach in less than a decade. Once achieved, this level of speech recognition will enable the development of many applications—such as transcription of multiperson conversations and meetings—that lie beyond the capabilities of today's speech recognition systems.

Acknowledgment

We thank the members of the Telephony Speech Algorithms Group for their comments.

References

1. S. Davis and P. Mermelstein, "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences," *IEEE Trans. Acoustics, Speech, and Signal Processing*, Aug. 1980, pp. 357-366.
2. H. Hermansky, "Perceptual Linear Prediction of Speech," *J. Acoustic Soc. America*, Apr. 1990, 1738-1752.
3. F. Jelinek, *Statistical Methods for Speech Recognition*, MIT Press, Cambridge, Mass., 1999.
4. G. Saon et al., "Maximum Likelihood Discriminant Feature Spaces," *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing* (ICASSP 2000), IEEE Press, Piscataway, N.J., 2000, vol. 2, pp. 1129-1132.
5. L. Mangu, E. Brill, and A. Stolcke, "Finding Consensus among Words: Lattice-Based Word Error Minimization," *Proc. Eurospeech 99*, http://www.edu/publications/pubs/EURO99.pdf.
6. J.L. Gauvain and C.H. Lee, "Maximum a Posteriori Estimation for Multivariate Gaussian Mixture Observations of Markov Chains, *IEEE Trans. Speech and Audio Processing*, Apr. 1994, pp. 291-298.

7. C.J. Legetter and P.C. Woodland, "Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models," *Computer Speech and Language*, vol. 9, p. 171, 1995.
8. S. Wegmann et al., "Speaker Normalization on Conversational Telephone Speech," *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing* (ICASSP 96), IEEE Press, Piscataway, N.J., 1996, pp. 339-343.
9. M.J.F. Gales, "Maximum Likelihood Linear Transforms for HMM-Based Speech Recognition," *Computer Speech & Language*, vol. 12, 1998, pp. 75-90.
10. J.G. Fiscus, "A Post-Processing System to Yield Reduced Word Error Rates: Recogniser Output Voting Error Reduction (ROVER)," *Proc. IEEE ASRU Workshop*, IEEE Press, Piscataway, N.J., 1997, pp. 347-352.
11. R. Lippmann, "Speech Recognition by Machines and Humans," *Speech Communication*, vol. 22, no. 1, 1997, pp. 1-15.

Mukund Padmanabhan *was the manager of the Telephone Speech Algorithms Group in the Human Language Technologies Group at the IBM T.J. Watson Research Center. He is now with Renaissance Technologies Inc. Contact him at mukund@rentec.com.*

Michael Picheny *is manager of the Speech and Language Algorithms Group in the Human Language Technologies Group at the IBM T.J. Watson Research Center. His interests encompass all aspects of speech recognition systems. Picheny received a PhD in electrical engineering from the Massachusetts Institute of Technology and is a Fellow of the IEEE. Contact him at picheny@us.ibm.com.*