A large-vocabulary continuous speech recognition system for Hindi

M. Kumar
N. Rajput
A. Verma

In this paper we present two new techniques that have been used to build a large-vocabulary continuous Hindi speech recognition system. We present a technique for fast bootstrapping of initial phone models of a new language. The training data for the new language is aligned using an existing speech recognition engine for another language. This aligned data is used to obtain the initial acoustic models for the phones of the new language. Following this approach requires less training data. We also present a technique for generating baseforms (phonetic spellings) for phonetic languages such as Hindi. As is inherent in phonetic languages, rules generally capture the mapping of spelling to phonemes very well. However, deep linguistic knowledge is required to write all possible rules, and there are some ambiguities in the language that are difficult to capture with rules. On the other hand, pure statistical techniques for baseform generation require large amounts of training data that are not readily available. We propose a hybrid approach that combines rule-based and statistical approaches in a two-step fashion. We evaluate the performance of the proposed approaches through various phonetic classification and recognition experiments.

1. Introduction

An automatic speech recognition (ASR) system consists of two main components—an acoustic model and a language model. The acoustic model of an ASR system models how a given word or "phone"[1] is pronounced. In most of the current ASR systems, the probability of a phone being spoken is modeled, using Baye's theorem, as follows:

$$P(M|\mathbf{O}) = \frac{P(\mathbf{O}|M)*P(M)}{P(\mathbf{O})}, \quad (1)$$

where $\mathbf{O}$ is the observation vector and M is the particular phone or word being hypothesized. Often, the probabilities $P(M)$ are assumed to be equal for all of the phones; hence, the term $P(\mathbf{O}|M)$ is used to compute the likelihood of the hypothesized phone. The acoustic model consists of the speech signal features to be used for $\mathbf{O}$, and a pattern-matching technique to compare these features against a set of predetermined patterns of these features for a given word or phone. Mel-Frequency Cepstral Coefficients (MFCC) are the most commonly used features for ASR. They represent the spectral envelope of the speech signal on the mel-frequency scale, which is dependent upon the particular sound being spoken. Hidden Markov model (HMM) and neural network (NN) are the most common techniques for acoustic modeling of ASR systems. We use HMMs based on allophones (context-dependent phones) in our ASR system. These HMMs model the output probability distribution (the probability of generating different values of MFCC in a given allophone state) and the transition probability (the probability of transition from one allophone state to another). At the time of speech recognition, various words are hypothesized against the speech signal. To compute the likelihood of a given word, the word is broken into its constituent phones, and the likelihood of the phones is computed from the HMMs.

[1] The term *phone* represents a basic unit of speech, a speech sound considered as a physical event. A word consists of one or more phones.

0018-8646/04/$5.00 © 2004 IBM

The combined likelihood of all of the phones represents the likelihood of the word in the acoustic model.

The language model of an ASR system predicts the likelihood of a given word sequence appearing in a language. The most common technique used for this purpose is an *N*-gram language model. An *N*-gram model provides the probability of the *N*th word in a sequence, given a history of $N - 1$ words—that is, $P(W_i | W_{i-1}W_{i-2}..W_{i-N+1})$. The *N*-gram model is trained over a large text corpus in the given language to compute these probabilities. For a hypothesized word, the language model score and the acoustic model score are combined to find the final likelihood of the word.

By using both the acoustic model and the language model, the combined likelihood of the word is computed as follows:

$$P(W) = P(\mathbf{O} | W_i) * P(W_i | W_{i-1}W_{i-2}..W_{i-N+1}). \quad (2)$$

For isolated word recognition, the above likelihood is computed for all words being considered, and the word having the highest likelihood is chosen as the recognized word. In the case of continuous speech recognition, the likelihood of a word is combined with the likelihood of other words to compute the combined likelihood of the sentence being hypothesized.

To train the acoustic model, a phonetically aligned speech database is required. However, acoustic models are required in order to automatically align a speech database. Hence, it becomes a chicken and egg problem. One possible method is to manually align the speech database; however, manually aligning a large speech database is very time-consuming and error-prone. Obtaining initial phone models for a new language is thus a challenging task. In [1], Byrne et al. have suggested techniques to create phone models for languages which do not have a lot of training data available. They have used knowledge-based and automatic phone mapping methods to create phone models for the target language, using phone models of other languages. Previous approaches [2, 3] to generate initial phone models include bootstrapping from a multilingual phone set and the use of codebook lookup. A codebook specifies the mapping to be used while performing the bootstrapping. The generation of this codebook requires linguistic knowledge of the languages. The technique mentioned in [2] requires a system already trained in the languages. On the other hand, the method in [3] requires labeled and segmented data in the language for which the system is to be trained. Authors in [4] describe various methods of generating the Chinese phone models by mapping them to the English phone models. This requires the collection of specific utterances of isolated monosyllabic data that is difficult for a language such as Hindi. Moreover, it may not be the best means for initializing the phone models that are to be used in large-vocabulary continuous speech recognition tasks. Cross-lingual use of recognition systems is also seen in [5], where the aim is to generate a crude alignment of words that do not belong to the language of the recognition system.

In this paper, we propose an approach for building good initial phone models through bootstrapping. We make use of the existing acoustic models of another language for bootstrapping. Following the approach proposed in [1], we define a phone mapping between the two languages to obtain an initial alignment of the target language speech data. However, in the case of Hindi, we have special acoustic classes, e.g., nasalized vowels and stressed plosives, which require more than one phone from the base language (English) for bootstrapping. We use this aligned data to obtain initial phone models of the target language. While segmenting the aligned data for target language phones, we use a module called a lexeme context comparator, which helps in differentiating phones in the target language which were mapped to same phone in the base language. The proposed approach requires relatively lower amounts of speech data for the new language to build initial phone models.

The second technique presented in this paper relates to baseform generation. For training the acoustic model, baseforms for the training words are required along with the initial phone models. These baseforms are also required during recognition for each word in the vocabulary. Since generating baseforms manually for large vocabularies is a time-consuming process, automatic baseform builders are important in all speech recognition applications.

Researchers have used a pure rule-based technique for baseform builders for phonetic languages [6]. The advantage of this technique is that once all of the rules are accounted for, the accuracy is very high; however, this requires deep linguistic knowledge that may be difficult to obtain [7]. While pronunciation rules can be extracted from existing online dictionaries, existing online dictionaries for Hindi are not exhaustive in their word coverage or on pronunciations. Additionally, each such online dictionary for Hindi requires a specific format in which the Hindi characters are encoded, thus making them even more difficult to use. It is easy to capture the general linguistic nature of phonetic languages, but their idiosyncrasies and exceptions are difficult to capture by rules. For example, in Hindi, deletion of the "schwa"[2] is very difficult to capture with rules [7]. The colloquial use of the language develops ambiguities that are too frequent to ignore in a speech recognition system. Such ambiguities are also difficult to capture by rules. On the other hand, using pure statistical techniques requires a large amount of training data that is not easily available for a new

[2] A *schwa* is a neutral middle vowel which occurs in unstressed syllables; it is represented by the /AX/ phone in our phone set.

language. Different statistical approaches have been tried for baseform builders. Decision trees [8–11], machine-learning techniques [12], delimiting, and dynamic time warping (DTW) [13] are a few of the techniques that have been studied. All of the statistical techniques require a large amount of training data for respectable accuracy. Moreover, their performance is compromised for "unknown words," typically proper nouns [9]. In order to improve the statistical techniques, other knowledge sources such as acoustics are used in conjunction with the spellings to obtain better results [14]. Pure acoustic-based baseform builders have also been built [15]. However, the techniques that use acoustics are restricted in their usage, since they require a recognition engine for the language and are better used for generating speaker-dependent pronunciations.

In this paper we present a hybrid approach that combines rule-based and statistical techniques in a novel two-step fashion. We use a rule-based technique to generate an initial set of baseforms and then modify them using a statistical technique. We show that this approach is extremely useful for phonetic languages such as Hindi. A detailed description of the pronunciation aspects of Hindi is presented in Section 3. The phonetic nature of the language can be exploited to a greater extent by using the rule-based approach, while the statistical technique can be used to improve on this. We experimented with two different techniques as the statistical component of our hybrid system—one of them uses modification probabilities, while the other uses context-dependent decision trees.

The rest of the paper is organized as follows. In Section 2, we describe our approach for bootstrapping the initial phone models. Our approach for a hybrid baseform builder is described in Section 3. The experiments conducted to evaluate the performance of the two approaches are presented in Section 4. Results corresponding to the experiments are discussed in Section 5, and we conclude in Section 6.

2. Bootstrapping of phone models

In the bootstrapping approach, an already existing acoustic model of a speech recognition system for a different language is used to obtain initial phone models for a new language. In the literature [2, 4], there are primarily two approaches used for bootstrapping. We explain these approaches using English as the base language and Hindi as the new or target language:

- *Bootstrapping through alignment of target language speech data* In the first approach, phonetic transcription of the target language text is written using the phone set of the base language. This is achieved by using a mapping defined between the two phone sets, which is detailed in the subsection on phone set mapping. The speech data in the target language is aligned using the speech recognition system of the base language. Initial phone models for the target language can then be built from the aligned speech data. The Hindi phone set is presented in **Figure 1**. For example,

  BHARAT –/BH AA R AX TX/ (actual);
  BHARAT –/B AA R AX TH/ (using English phone set).

  In this case, the phones /BH/ and /B/ in the target language are both mapped to phone /B/ in the base language. Hence, to initially obtain the aligned data for /BH/, the data aligned with /B/ is randomly distributed between /BH/ and /B/. Phone /TX/ in the target language is mapped to phone /TH/ in the base language.
- *Bootstrapping through alignment of base language speech data* In the second approach, speech data of the base language itself is aligned using its speech recognition system. The aligned speech data of the base language is used as the aligned speech data for the target language using the mapping between the two phone sets. For example,

  BAR – /B AA R/.

  The aligned data for /B/ is randomly distributed to obtain the aligned data for /BH/ and /B/.

Proposed approach

We have proposed a new technique for bootstrapping which provides more accurate initial phone models for the target language. We have modified the first approach as described above, so that the aligned speech data for two similar phones in the target language can be easily separated, for example for phones /BH/ and /B/. We propose to use both the phone sets, i.e., the phone sets of base and target languages, to avoid the confusion between the phones in the target language which are mapped to the same phone in the base language.

Figure 2 shows the technique that is used to align Hindi speech by using an English speech recognition system. A mapping $h(\,)$ from a Hindi phone set denoted by Γ to an English phone set denoted by Π is used to generate the pronunciation of Hindi words by the English phone set. Using linguistic knowledge, this mapping is based on the acoustic closeness of the two phones. The mapping is such that each phone $\gamma \in \Gamma$ is mapped to one and only one phone in Π. A vocabulary created by such a mapping is used to align Hindi speech data. Since more than one element in Γ may map to a single element in Π, $h(\,)$ is a many-to-one mapping in general and hence cannot always be used in reverse to obtain γ from π. Therefore, in order to recreate the alignment labels with Hindi phones, an inverse mapping $h^{-1}(\,)$ will not be feasible. A lexeme context comparator is used to generate the correct labels

Hindi phone (γ)	Hindi alph	$h(\gamma)$	$\rho(\gamma)$	Hindi phone (γ)	Hindi alph	$h(\gamma)$	$\rho(\gamma)$	Hindi phone (γ)	Hindi alph	$h(\gamma)$	$\rho(\gamma)$	Hindi phone (γ)	Hindi alph	$h(\gamma)$	$\rho(\gamma)$
AA	आ	AA	AA	DDN	ङ	DD	DD+R	JH	ज	JH	JH	S	स	S	S
AAN	आ	AA	AA+N	DH	ढ	DH	DH	JHH	झ	JH	JH+HH	SH	श	SH	SH
AE	ऐ	AE	AE	DHH	घ	DH	DH+H H	K	क	K	K	T	ट	T	T
AEN	ऐं	AE	AE+N	DN	ण	DX	DX+N	KD	क्	KD	KD	TD	ट्	T	T
AW	औ	AW	AW	DXH	ढ	DX	DX+H H + R	KH	ख	KD	KD+H H	TH	थ	TH	TH
AWN	औं	AW	AW+N	DXX	ढ़	DX	DX+H H	L	ळ	L	L	THH	ठ	TH	TH+H H
AX	अ	AX	AX	EY	ए	EY	EY	M	म	M	M	TX	त	TH	TH
AXN	अं	AX	AX+N	EYN	एं	EY	EY+N	N	न	N	N	UH	उ	UH	UH
B	ब	B	B	F	फ	F	F	NG	ङ	NG	NG	UHN	उं	UH	UH+N
BD	ब्	BD	BD	G	ग	G	G	OW	ओ	OW	OW	UW	ऊ	UW	UW
BH	भ	BD	BD+HH	GH	घ	GD	GD+H H	OWN	ओं	OW	OW+N	UWN	ऊं	UW	UW+N
CH	च	CH	CH	HH	ह	HH	HH	P	प	P	P	V	व	V	V
CHH	छ	CH	CH+HH	IH	इ	IH	IH	PD	प्	PD	PD	Y	य	Y	Y
D	ड	D	D	IY	ई	IY	IY	PH	फ	P	PD+H H	Z	ज़	Z	Z
DD	ड्	DD	DD	IYN	ईं	IY	IY+N	R	र	R	R				

Figure 1

Hindi phonemes for characters in Hindi. Mappings are shown using an English phone set.

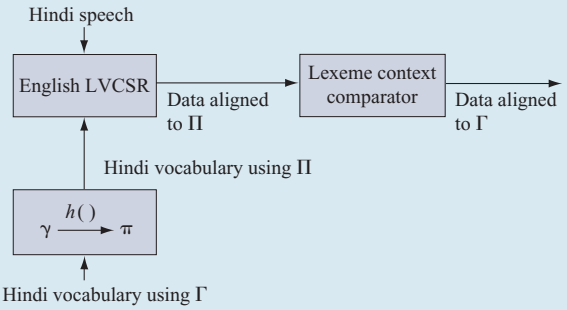

Figure 2

Alignment of the target language data. (LVCSR: large-vocabulary continuous speech recognition.)

from $\pi \in \Pi$. This uses the context to resolve the ambiguity which arises from the one-to-many mapping $h^{-1}(\ )$. To illustrate the requirement of a lexeme context comparator, we take the example of two Hindi words, भारत and बहुत. The baseforms for these words are shown in **Table 1**. For both words, the alignment would be generated for the phone /B/. However, this /B/ must be replaced by /BH/ if the word is भारत and by /B/ if the word is बहुत. This information is not available by using the mapping $h^{-1}(\ )$. Therefore, a lexeme comparator is used to examine the lexemes of the words and disambiguate for such cases.

The algorithm can be stated in the steps mentioned below:

1. For a feature vector labeled with a phone $\pi \in \Pi$, form a subset $\Psi \subset \Gamma$ using the inverse mapping $h^{-1}(\ )$ [since $h^{-1}(\ )$ is a one-to-many mapping in general].
2. If Ψ is a singleton, change the label of the feature vector to the element $\gamma \in \Psi$.
3. If not, from the lexeme context of the feature vector, compare the two phonetic spellings of the two lexemes (one written with phones in Π and other with phones in Γ) to which this vector belongs. Using this information, handle the disambiguity and choose the phone from Ψ that satisfies the mapping $h^{-1}(\ )$ for the lexeme—for example, /B/ and /BH/.

This technique would generate the aligned Hindi speech corpus without the need for a Hindi speech recognizer. Although this alignment may not provide exact phone

boundaries, it would serve the purpose of building the initial phone models. The inaccurate phone boundaries are a result of phonetic space differences in the two languages owing to the different acoustic characteristics of the languages. This depends on the two languages; if the languages are acoustically similar, we can have accurate phone boundaries using the above technique. It should be noticed that using the phone set of the target language in the lexeme context comparator not only separates the aligned data for /B/ and /BH/ but also provides the right context information for other phones in the aligned speech corpus. This context information would otherwise have been abused because of the many-to-one phone mapping from target language to base language.

Phone set mapping

The International Phonetic Association (IPA) [16] has defined phone sets for labeling speech databases for sounds of a large number of languages, including Hindi. However, there are some sounds in Hindi which are not included in the IPA phone set but are important when building phone models that are to be used for the purpose of automatic speech recognition. In continuous speech recognition tasks, the purpose of defining a phonetic space is to form well-defined, non-overlapping clusters for each phoneme in the acoustic space. This clustering makes it easier for the system to recognize the phone to which an input utterance of speech belongs. For the same number of data and phoneme models, a better phone set is one that gives a higher classification rate and is able to distinguish the words present in the vocabulary of the language. We define a Hindi phone set which can cover all the different sounds that occur in Hindi. This phone set takes into consideration the fact that even though Hindi is a phonetic language, from an acoustic point of view some phones such as plosives have different acoustic properties when they occur at the end of the word. Taking these into account, we have constructed a Hindi phone set consisting of 61 phones (including the inter-word silence *D*$ and long pause silence *X*) to represent the sounds in Hindi. It is seen that of these 61 phones, 39 are already present in English. Figure 1 shows the corresponding characters as written in Hindi script. In the figure, $h(\gamma)$ represents the mapping of Hindi phones to the corresponding English phones for aligning the Hindi data using English acoustic models, and $\rho(\gamma)$ represents the mapping to obtain the initial phone models for the Hindi phones from English data. In addition to ten English vowels, Hindi has nine nasalized vowels (AAN, AEN, AWN, AXN, EYN, IYN, OWN, UHN, UWN). Each plosive phone (B, D, K, P, T) has an additional phone (BD, DD, KD, PD, TD) to represent the acoustic dissimilarity when they occur at the end of a word. The bootstrapping approach described in the preceding subsection requires a mapping from the phones of the base language to the phones of the target language. A phone set mapping is defined using the linguistic knowledge of the two languages. We define three categories of mapping as follows:

Table 1 Baseforms for two Hindi words.

Hindi word	*Hindi baseform*	*English baseform*
भारत	BH AA R AX TD	B AA R AX TD
बहुत	B AX HH UH TD	B AX HH UH TD

- *Exact mapping* Some of the phones may be common to both the base and the target language. For example, many vowels such as /AX/, /AA/, and /IY/ are common to English and Hindi, and they have an exact mapping from one language to the other. The mappings $h(\,)$ and $\rho(\,)$ are the same for such phones.
- *Merging* Some of the phones in the target language may have sounds from more than one phone in the base language. For example, Hindi has some nasalized vowels such as /AAN/ and /EYN/, which are a combination of the corresponding vowel and nasal sound /N/. For these phones, one-to-many mapping is defined from such Hindi phones to their English counterparts. For example, the Hindi phone /GH/ is a combination of the English phones /GD/ and /HH/ while creating the mapping $\rho(\,)$. The mapping for such phones differs in the case of $h(\,)$ and $\rho(\,)$.
- *Approximation* Some of the phones in the target language may not be present in the base language at all. Such phones are simply mapped to the closest phone in the base language. For example, phone /TX/ in Hindi (भारत—BH AA R AX TX) is mapped to phone /TH/ in English (B AA R AX TH). The mappings $h(\,)$ and $\rho(\,)$ are the same for such phones.

Refining phone set mapping

We now present a method that is used to improve the initial phone set mapping $\rho(\,)$. This method is based on a measure of phonetic similarity between the phones in Γ and the phones in Π. One possible measure of similarity is the distance between the phones in the MFCC domain. Each phone of Π is modeled by a normal distribution, and the phonetic distance of a phone $\gamma \in \Gamma$ from a phone $\pi \in \Pi$ is defined as

$$D(\gamma, \pi) = \frac{\sum_{v_i \in \gamma} (v_i - m_\pi)^2}{\|\Gamma\|},$$

where v_i represents a 24-dimensional MFCC vector belonging to γ and m_π is the mean vector corresponding

to π. However, we used a distance measure based on the log likelihood of the phone models in Π for each test vector in $\gamma \in \Gamma$. The mean of log likelihoods is taken as the measure of acoustic similarity between the phones in the two languages. This measure is calculated for each phone $\gamma \in \Gamma$ over all of the phones in Π that are considered to be close to γ. The mapping $\rho(\,)$ is refined if the acoustic similarity measure shows that a phone γ is closer to some phone π' than it is to π, to which it was initially mapped. The log-likelihood-based distance measure produces better results. As a result of the refinement, we changed the mapping of /DDN/ from /DD + HH/ to /DD + R/ and of /DXH/ from /DD + HH/ to /DD + HH + R/.

In Section 4, we describe the phonetic classification experiment which illustrates the improved performance of the initial phone models that have been discussed in this section.

3. Hybrid baseform builder for phonetic languages

We present a technique for generating baseforms for phonetic languages such as Hindi. As is inherent in phonetic languages, rules generally capture the spelling-to-phoneme mapping very well. However, deep linguistic knowledge is required to write all of the possible rules, and there are some ambiguities in the language that are difficult to capture with rules. On the other hand, pure statistical techniques for baseform generation require a large amount of training data, which is not readily available. We propose a hybrid approach that combines rule-based and statistical approaches in a two-step fashion. We evaluate the performance of the proposed approaches through various phonetic classification and recognition experiments.

Issues specific to Hindi

Hindi is a phonetic language, which implies that there is generally a strong correlation between its written and spoken form. It has adopted various Arabic and Persian words which introduce characters that are pronounced differently by different speakers, e.g., फ़–फ and ज–ज़. Hindi also has a few distinct phones which are characterized by more than one sound being spoken simultaneously. For such phones, such as stressed plosives (/DXH/, /DXX/, and /DHH/) and nasalized vowels, acoustic data from multiple phones is required for bootstrapping. In written Hindi, each consonant is associated with an inherent schwa, which is not explicitly presented. Other vowels are overtly written diacritically or nondiacritically around the consonant. Depending on the context, the schwa is at times absent, resulting in an implicit stop, as explained in the subsection on limitations of rule-based techniques. Contexts which lead to the deletion of the schwa require deep linguistic knowledge. For example, written Hindi has a special characteristic of half-consonants. These are the consonants without the schwa sound discussed in the subsection on rule-based baseform generation.

Statistical baseform generation

Many statistical techniques have been tried for baseform builders, as mentioned in Section 1. The statistical approach that we have used is based on context-dependent decision trees [17]. In this approach, a tree is built for each letter. Training a tree for a particular letter involves partitioning the training data (corresponding phone or phone sequence) into several leaf nodes, depending on the letter and phone context. This training data represents letter-to-phone or phone sequence mappings for all words in the dictionary. The partitioning is achieved by splitting the data at each node into two subnodes which are maximally heterogeneous. Heterogeneity between two nodes is defined as the difference in the number of occurrences of a given phone or phone sequence. We stop the partitioning when the heterogeneity between the two subnodes is less than an empirically decided threshold value, or when the size of the data at the node is less than an empirically decided threshold value. The phonetic context comprises five previous phones, and the letter context is specified by five previous and five succeeding letters. The set of phonetic questions is mentioned in the subsection on initial phone models, while the questions on letter context are of the form "Is the letter at context position +1 'b' ?" Such questions are used to partition the data. Once such a tree is built, leaves of each tree specify a probability distribution for letter-to-phone mapping for a particular phonetic and letter context. Generating baseforms from these context-dependent trees involves traversing the tree for each letter and generating the baseform for the input word. The performance of the statistical approach is described in Section 5.

Generation of rule-based baseforms

Specifying rules to build baseforms from input spelling works for a large number of words for phonetic languages. The knowledge of phone sets and pronunciations of each phone, along with the linguistic knowledge of the language, is used to specify rules that convert spellings to sounds. Rules are of the form that a given letter and its context in a word are mapped to a particular phoneme sequence. The Hindi phonetic character set that we have used is described in detail in [18]. All of the 33 consonant characters in written Hindi also have a corresponding representation as half-consonants. The only difference between the sounds of the consonants and the corresponding half-consonants is that the former almost always have the sound of the vowel /AX/ present in them.

The half-consonants have just the sound of that phone. Appending the schwa sound to the sounds of their corresponding half-consonants generates the sounds for consonants; for example, ब–ब् are a corresponding consonant and half-consonant pair. The rules that we have used are a simple mapping of these consonants to their corresponding consonant sounds.

Incorporating redundancy through parallelism

Using the mappings of characters to phones, we have built a rule-based baseform builder for the Hindi language. However, the mappings have to be one-to-many in order to generate alternate pronunciations. In Hindi, multiple pronunciations exist for two reasons:

1. Though the language has specific pronunciations for each literal and it does not change with the context, people often speak a character differently (most common pairs are फ़–फ, ज़–ज, and स–श). This is because Hindi has adopted various Arabic words that are pronounced differently by different speakers.
2. Hindi is often erroneously written; characters may be interchanged, the most common being ड–ड़ and ढ–ढ़.

To handle these statistically significant mispronunciations (misspellings), rules must be modified to build multiple baseforms whenever such characters are encountered. Thus, we build parallel baseforms for all words that have these characters, creating a saturated baseform vocabulary which is a superset of the true baseform vocabulary. This increases the size of the baseform vocabulary, and hence the search time also increases during decoding. However, better acoustic scores are expected when the desired lexeme is present in the baseform vocabulary than when only one baseform is available for each word. In Section 5, we see the effect of parallel baseforms on the recognition accuracy and also on the search time.

Limitations of rule-based techniques

As mentioned in the previous subsection, we need alternate baseforms to capture the varied pronunciations required for a speech recognition task. However, since these rules have to be made to include all contexts for a character, they incorporate redundancy in the generated baseforms. Thus, in order to prune these redundant cases, we need a statistical technique to differentiate the contexts where the parallelism generated is redundant and where it is useful. Also, though the structure of Hindi is phonetic, it has certain implicit stops that render the phonetic spelling not completely obvious from the word spelling. This can be illustrated by the example of the two Hindi words उसने and सन. The rule-based baseform builder would generate the phone /UH/ corresponding to

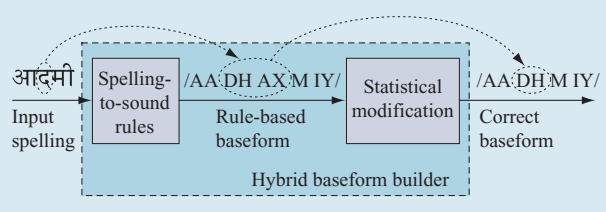

Figure 3

Hybrid baseform builder framework.

the vowel उ, /S AX/ corresponding to the consonant character स, a phone /N/ corresponding to the consonant न, and /EY/ corresponding to ए. This would give the phonetic spelling of उसने as / UH S AX N EY /, and that for सन would be /S AX N/. The former spelling should actually have been / UH S N EY /. The implicit stop in स is not reflected in उसने, since it is actually pronounced as उस्ने. To capture such variations in similar character sequences, we train a statistical model that determines the absence of implicit stops after consonants and hence makes corresponding changes to the rule-based baseform generated earlier. This is detailed in the next section.

Framework for a hybrid baseform builder

As illustrated in **Figure 3**, the input to a hybrid baseform builder is the spelling of the word. A rule-based system is used to generate all possible baseforms for this word. The phonetic structure of the language is captured by the rule-based system. The rules used are fairly simple and are easy to derive without deep linguistic knowledge. In the second step, the baseforms generated by this rule-based system along with the spelling are input to a statistical baseform modification system. We define a set of unruly phones for the language which comprises phones for which the parallelism incorporated is redundant in certain cases or for which the rules are too complex to be derived without deep linguistic knowledge. Thus, the statistical technique takes care of the complex rules and the ambiguities. Since we capture only the complex rules and the ambiguities by the statistical approach, we do not require a large body of training data, but only the data specific to these phones. Moreover, since we are using the statistical technique over the rule-based baseforms, we have a richer context to train the model (left and right phone context) than in previous approaches [14] that learned letter-to-sound mappings using the left and right letter context and *only* left phone context. The output of this statistical system is the baseform set for the input word.

As shown in Figure 3, spelling-to-sound rules are used to generate the initial baseform for the input word आदमी.

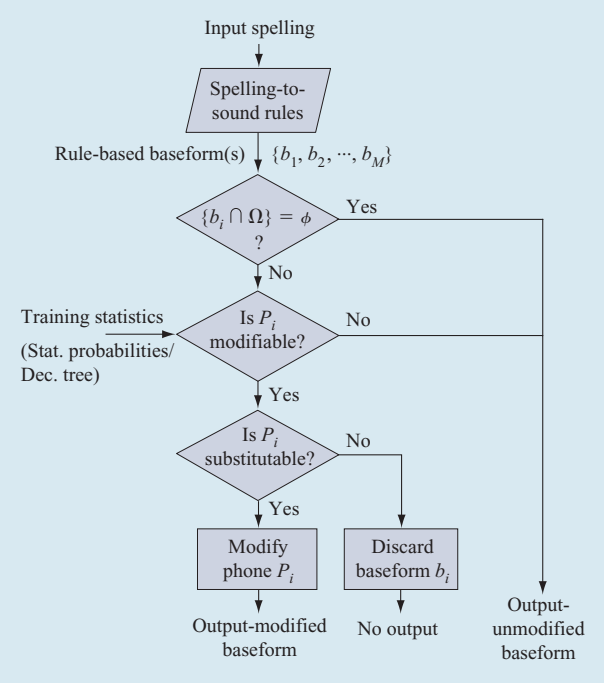

Figure 4

Context-dependent statistical technique to modify rule-based baseforms.

In the next step, statistical modification is applied over only the /AX/ (schwa) phone to modify the rule-based baseform. The deletion of the /AX/ phone requires complex rules in Hindi, as explained in the subsection on limitations of rule-based techniques. Thus, we capture a complex rule of Hindi using statistics in our hybrid approach.

Probabilistic baseform modification

We use statistical techniques to modify the baseforms generated by a rule-based technique. During training, given the baseforms generated by rule-based techniques and the true baseform vocabulary, we learn which alternate baseforms are redundant and which baseforms are to be modified. In this section we show how the context of the unruly phone in a baseform can be used to learn about the correctness or existence of that particular phone in the baseform.

If the phone set $P_1, P_2, \cdots, P_N$ denoted by Γ has a subset of unruly phones $P_{u1}, P_{u2}, \cdots, P_{uK}$ (phones that do not have an exact spelling-to-sound rule) denoted by Ω, we can build statistical techniques to modify those rule-based baseforms that have these phones present in them. The first step in using statistical training for improving the performance of a rule-based baseform builder is to identify the unruly set Ω. For Hindi, these phones are AX, PH, F, JH, and Z. The aim is to use information on the context of these phones when they appear in the baseform of a word generated by rule-based techniques. Depending on the context, as illustrated in **Figure 4**, the baseform is not modified at all; only the phone under consideration is modified in the baseform, or the baseform itself is discarded. To train such a system for baseform correction, for each phone P_{ui} in the unruly set Ω, the following steps are followed:

1. Build a training set of baseforms by manually correcting the baseforms in which the phone P_{ui} appears.
2. Record five previous and five succeeding phones $\{P_{c-5}, P_{c-4}, P_{c-3}, P_{c-2}, P_{c-1}, P_{c1}, P_{c2}, P_{c3}, P_{c4}, P_{c5}\}$ of P_{ui} to obtain its context. If the context exceeds the baseform length, include X as the context phone at those positions.
3. Create two sets of context tables, one which corresponds to contexts in which the phone P_{ui} would remain in the correct baseform, and the second set in which the phone P_{ui} should be modified to make the baseform correct.
4. Assign modification probabilities to the phone set $\{P_1, P_2, \cdots, P_N\}$ for each context location by counting the number of occurrences of the phones in the context tables.

Using this training procedure, all conditional probabilities of modification for all contexts and for all phones in the subset are calculated. Therefore, for each phone P_{ui},

$$p_{kj} = \text{probability of modification of the phone } P_{ui} \text{ when the phone at context position } k \text{ is } P_j, \quad (3)$$

where $k = -5, .., 5$ and $j = 1, 2, .., 61$.

At the time of baseform modification, when a new baseform having a phone P_{ui} is encountered, the training probabilities are used to find the score s_{ui} of modification of the phone given the complete context of phone P_{ui} in this baseform, using the weighted sum as

$$s_{ui} = \sum_{k=-5}^{5} w_k \left(\sum_{j=1}^{61} \phi_j p_{kj} \right), \quad (4)$$

where $\phi_j = 1$ when the phone at position k is P_j, else $\phi_j = 0$; w_k is the weight at the context position k; and $\Sigma_{k=-5}^{5} w_k = 1$, with a position nearer to the phone in question being given a higher weight.

If the calculated score is higher than an empirically chosen threshold, the baseform is modified, or else it is left unchanged. As shown in Figure 4, modifications can be of two types, depending on the phone P_{ui}. Either the

Table 2 Context details and modification results.

Hindi word	*Baseform*	*Unruly phone*	*Context* −5 −4 −3 −2 −1 1 2 3 4 5	*Modification suggested*
आदमी	AA DH AX M IY	AX	X X X AA DH M IY X X X	Delete phone AX
दम	DH AX M	AX	X X X X DH M X X X X	Don't modify
जहाज़	Z AX HH AA Z	Z	X X X X X AX HH AA Z X	Delete baseform

baseform itself is discarded, being judged as a redundant alternate baseform, or the phone P_{ui} is deleted to correct the baseform.

As shown in **Table 2**, the baseforms for words in the first column are checked for the presence of unruly phones. The third column shows the unruly phone in the corresponding baseform. The context of this unruly phone is extracted (as shown in the fourth column); on the basis of the probabilistic modification for this context, a decision is taken to delete the unruly phone, to delete the baseform, or to leave the baseform unchanged. This method of first generating baseforms using the spelling-to-sound rules and then using the contexts in these baseforms to modify them has the advantage that we need training data only for baseforms that have the phones in the unruly set Ω in them. Thus, less training data is required; also, use of the statistical training mentioned above corrects most of the baseforms, as shown in Section 5.

Decision trees for baseform modification

Counting the number of occurrences of phones in the contexts is not the best way to estimate p, because there are ten contexts and 61 phones in Hindi that create 610 different positions to search for the context-based decision. In this section we illustrate the use of decision trees for determining the modifications to be done to baseforms having phones in the unruly set Ω. A decision tree is built for each phone in the unruly set. Creating such decision trees involves asking questions to a set of training baseforms. These questions partition the set of baseforms at each node into the best possible context that would differentiate between the modifiable and nonmodifiable baseform. For each phone P_{ui} in the unruly set Ω, we store the rule-based baseforms and correct baseforms that have the phone P_{ui}. Next we describe how questions are selected at each node.

Best question selection criterion

The best questions at any node are the one that divides the data into two sets of nearly the same size and the one whose sets differ most in terms of baseforms that have the phone to be modified and phone not to be modified. The set of questions that we use are the ones being used in [13]. Each question is of the type "Does the phone at position −3 belong to the subset P, PH, B, BH, M?" All questions result in a binary yes/no answer, and each node correspondingly has two children. To decide the best question at any node in the tree, we use the following score:

$$s = \frac{\|m_Y - m_N - u_Y + u_N\|}{m_Y + m_N + u_Y + u_N}, \tag{5}$$

where m_Y is the number of cases in which a question results in a yes answer and the phone has to be modified; m_N is the number of cases in which a question results in a no answer and the phone has to be modified; u_Y is the number of cases in which a question results in a yes answer and the phone need not be modified; and u_N is the number of cases in which a question results in a no answer and the phone need not be modified. The question that gives highest score is chosen as the best question for that node.

Creating the tree

For all training baseforms at a root node, questions are asked and scores calculated using Equation (5). The best question is used to divide the data into two distinguishing sets. The process is continued until the following criterion of stopping is reached: A node is turned into a leaf when no question yields a score good enough to be constituted as intelligible or if the number of baseforms at the node is too small to divide. After the tree is built, each leaf represents a set of contexts that must be satisfied in order to reach it. Each leaf is marked either as modifiable or unmodifiable depending on the answer to the previous question.

Baseform modification

When a new baseform is presented to the system, the tree is traversed, and, on the basis of the context of the unruly phone in the baseform, a leaf node is reached. Each leaf node is marked as either modifiable or unmodifiable. If the leaf reached by traversing the tree is marked modifiable, the unruly phone for which the tree was

Table 3 Phonetic classification rates for Hindi data using the Hindi phone models created by the English data.

Hindi phonetic space method	*Hindi data labeling method*	*Classification rate (%)*
Context-based	Random	16.23
Random	Random	21.29
Random	Lexeme context	23.82
Modified with distance	Modified with distance	26.99

traversed is modified and the correct baseform is generated. Modifications in the baseform depend on the nature of the unruly phone present in the baseform. The phone can be deleted to make the baseform correct (in the case of AX), or the alternate baseform can be deleted (in the case of JH–Z, PH–F). On the other hand, if the leaf reached has been marked as unmodifiable, the input baseform is considered by the decision tree to be correct and is left unchanged.

4. Experiments

In this section we describe the experiments conducted to evaluate the performance of the proposed approaches. We use 24-dimensional Mel-Frequency Cepstral Coefficients (MFCC) as the feature vector of the speech data. To capture the dynamics of the speech signal, four previous and four succeeding MFCC vectors are concatenated to the current MFCC vector, and linear discriminant analysis (LDA) is applied on the concatenated vector to reduce the dimensionality of the feature vector from 24×9 to 60 dimensions. The vectors so obtained are used to model the output distribution of Hidden Markov Models (HMM). The acoustic models are trained over 200 hours of speech data collected from more than 500 speakers [19].

Initial phone models

We bootstrapped the initial phone model of the Hindi phone set, consisting of 61 phones from the phone model of the IBM U.S. English speech recognition system ViaVoice*, which has 52 phones. An initial phone set mapping was defined between the two phone sets using the approach described in Section 2. Using this mapping, the proposed bootstrapping approach described in Section 2 was used to obtain the aligned Hindi speech data. This data was used to refine the initial mapping.

The initial phone models so obtained were used to generate context-dependent phone models. Context-dependent trees are used to divide the phonetically aligned data. The context of a phone comprises five phones previous to and five phones succeeding the phone in consideration. Using the speech corpus, 3,718 context-dependent phones were built, and a set of 115 questions were used to build the context-dependent tree. An example of the questions asked could be "Does the phone at context position +1 belong to set { AX, AA, AE }?" Every question results in a yes or no answer. Data at the given node is split depending upon the answer to the question. The question selected as the best question gives the highest gain in likelihood after splitting. Splitting is stopped if either the number of vectors for a given node is less than a threshold or the likelihood gain from splitting the data of the node is less than a threshold. The leaves of the tree represent the phone with a particular context. The system generated a total of 3,718 context-dependent phones.

Hybrid baseform builder

Two experiments were conducted to measure the performance of the hybrid baseform builder. The first experiment measures the correctness of the generated baseforms, and the second experiment uses the generated baseforms in a Hindi speech recognition task. In both experiments, we compare the performance of the hybrid baseform builder with the pure rule-based approach and the pure statistical approach. Data preparation is the same for both experiments. Human experts have created a phonetic dictionary consisting of 12,350 Hindi words; of these, 11,510 words were used as the training set and 840 words were used to test the system. The rule-based baseforms were generated from the 11,510 training words. These baseforms, along with the true baseforms of the training set, were used to train the statistical component of the hybrid system for the five unruly phones mentioned in Section 3. The true baseforms of the training set of 11,510 words were used to train the pure statistical system. Tests were performed on the remaining 840 words. With test words as the input, four dictionaries were created using the four techniques of baseform generation: pure rule-based technique, hybrid technique with probabilistic modification, hybrid technique with decision-tree-based statistical component, and pure statistical system. Each of the four dictionaries contained different numbers of baseforms owing to the ability of the method to identify and discard redundant baseforms.

Measuring correctness of baseforms

In this experiment, we measure the correctness of the baseforms generated. The metric used for measuring the accuracy of the proposed approach is the baseform error rate. A baseform error occurs when the correct baseform is not present in the generated baseform vocabulary. The manually generated baseforms provided the standard for comparison in our experiment. The baseform vocabulary generated by the human experts for the 840-word test set consisting of 978 baseforms was compared with each of the four dictionaries.

Speech recognition experiment

Since one of the goals of the baseform builder is to generate baseforms that are used in a speech recognition system, the second experiment uses the generated baseforms in a recognition experiment. We have used the IBM speech recognizer for the Hindi language as the base recognition system [20]. This is a large-vocabulary, speaker-independent continuous speech recognition system. The Language Model has been trained on a text corpus of 20 million words that represents text from different domains. It consists of a trigram model with an open vocabulary and an unknown word probability of 0.00025. The vocabularies generated by the four techniques for the set of 840 words were used in the recognition experiment. The test set for the recognition experiment consisted of ten speakers, each with 200 continuous speech utterances of Hindi from this vocabulary of 840 words. This constituted a total of about three hours of speech. The baseform vocabulary created by human experts for these 840 words was used to compare the recognition accuracy with the four techniques.

5. Results

In this section, we present the results obtained for the various experiments which are described in the preceding section.

Phone models

Table 3 shows the results for phonetic classification of the Hindi data over the Hindi phonetic space generated through bootstrapping. Normally the phonetic classification rate is seen to be around 40–50% for most of the languages [3] with a trained system. The rate of 27% obtained for the Hindi language without using context-dependent models is a promising reason for using the phone models generated by the method described. The distance-measure technique provides an insight into the measure of closeness between the phone sets of the two languages. This is used to modify the mapping in order to create a better phonetic representation of the Hindi phones in the English data space. This modified mapping provides a 13% relative improvement in the rate of classification. Also, the use of lexeme context to label the Hindi data is a rapid way of generating the labeled data for a new language. Its advantage is reflected by an improved classification rate of 23.82% compared with no use of lexeme context information and random distribution of the data among phones that had a many-to-one mapping in $\rho(\,)$.

Table 4 Correct baseforms generated.

Baseform type	*Vocabulary size*	*Correct baseforms (%)*
Rule-based	1006	68.51
Probabilistic modification	1006	73.93
Decision-tree-based modification	912	80.47
Pure statistical	1698	85.38

Table 5 Recognition rates for the different vocabularies.

Baseform type	*Vocabulary size*	*Recovery rate (%)*	*Time (s)*
Rule-based	1006	81.97	3605
Probabilistic modification	1006	85.26	3228
Decision-tree-based modification	991	85.46	3218
Pure statistical	1698	83.29	3443
Correct	978	85.33	3152

Baseform builder

Baseform correctness experiment

Table 4 shows the improvement by using the statistical approach and decision trees over the rule-based baseforms. It is seen that the number of correctly generated baseforms increases and redundant baseforms are removed.

For an equivalent amount of training data, a completely statistical system gives better accuracy than our hybrid system; however, the generated baseform vocabulary size is considerably higher, as shown in Table 4. This increase in the size of baseform vocabulary for the statistical system has implications for the speech recognition task in that more decoding time is required.

Speech recognition experiment

Results in **Table 5** suggest that using the decision for modifying baseforms yields the highest recognition accuracy for the baseform builder. Moreover, owing to the reduction in the size of the baseform vocabulary, the time

taken for decoding is also reduced. It is observed that the hybrid technique can generate a smaller and yet better vocabulary from the rule-based baseforms. The baseform vocabulary created by human experts for these 840 words was used to compare the recognition results for vocabularies of the four techniques with that of the true baseforms. The last row of Table 5 specifies the recognition rates that are achieved for the manually generated baseform vocabulary.

6. Conclusion

In this paper we have presented two novel techniques that were used to build a continuous large-vocabulary Hindi speech recognition system. A new technique for bootstrapping the initial phone models has been presented. Another approach for a hybrid baseform builder was presented which can be used to automatically generate baseforms for phonetic languages. Recognition rates have been reported, and the improvement due to the proposed techniques has been highlighted.

*Trademark or registered trademark of International Business Machines Corporation.

References

1. W. Byrne, P. Beyerlein, J. M. Huerta, S. Khudanpur, B. Marthi, J. Morgan, N. Peterek, J. Picone, D. Vergyri, and W. Wang, "Towards Language Independent Acoustic Modeling," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Istanbul, 2000, pp. 1029–1032.
2. J. Kohler, "Multi-Lingual Phoneme Recognition Exploiting Acoustic–Phonetic Similarities of Sounds," *Proceedings of the International Conference on Spoken Language Processing*, Atlanta, 1996, pp. 2195–2198.
3. O. Anderson, P. Dalsgaard, and W. Barry, "On the Use of Data-Driven Clustering Technique for Identification of Poly- and Mono-Phonemes for Four European Languages," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Adelaide, Australia, 1994, pp. 121–124.
4. M. C. Yuen and P. Fung, "Adapting English Phoneme Models for Chinese Speech Recognition," *Proceedings of the International Conference on Spoken Language Processing*, Sydney, Australia, 1998, pp. 80–82.
5. T. A. Faruquie, C. Neti, N. Rajput, L. V. Subramaniam, and A. Verma, "Translingual Visual Speech Synthesis," *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, New York, 2000, pp. 1089–1092.
6. M. Choudhury, "Rule-Based Grapheme to Phoneme Mapping for Hindi Speech Synthesis," presented at the 90th Indian Science Congress of the International Speech Communication Association (ISCA), Bangalore, 2003.
7. M. Choudhury and A. Basu, "A Rule Based Schwa Deletion Algorithm for Hindi," *Proceedings of the International Conference on Knowledge Based Computer Systems*, Mumbai, 2002, pp. 343–353.
8. E. Fosler-Lussier, "Multi-Level Decision Trees for Static and Dynamic Pronunciation Models," *Proceedings of the Eurospeech Conference*, Budapest, 1999, pp. 459–462.
9. A. W. Black, K. Lenzo, and V. Pagel, "Issues in Building General Letter to Sound Rules," *Proceedings of the 3rd European Speech Communication Association (ESCA) International Workshop on Speech Synthesis,"* Australia, 1998, pp. 77–80.
10. A. K. Keinappel and R. Kneser, "Designing Very Compact Decision Trees for Grapheme-to-Phoneme Transcription," *Proceedings of the Eurospeech Conference*, Scandinavia, 2001, pp. 1911–1914.
11. J. Suontausta and J. Hakkinen, "Decision Tree Based Text-to-Phoneme Mapping for Speech Recognition," *Proceedings of the International Conference on Spoken Language Processing*, Beijing, 2000, pp. 199–202.
12. F. Mana, P. Massimino, and A. Pacchiotti, "Using Machine Learning Techniques for Grapheme to Phoneme Transcription," *Proceedings of the Eurospeech Conference*, Scandinavia, 2001, pp. 1915–1918.
13. R. W. P. Luk and R. I. Damper, "Inference of Letter–Phoneme Correspondences by Delimiting and Dynamic Time Warping Techniques," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, San Francisco, 1992, pp. 61–62.
14. L. R. Bahl, S. Das, P. V. deSouza, M. Epstein, R. L. Mercer, B. Merialdo, D. Nahamoo, M. A. Picheny, and J. Powell, "Automatic Phonetic Baseform Determination," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Toronto, 1991, pp. 173–176.
15. B. Ramabhadran, L. R. Bahl, P. V. deSouza, and M. Padmanabhan, "Acoustics-Only Based Automatic Phonetic Baseform Generation," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Seattle, 1998, pp. 309–312.
16. J. Wells and J. House, *The Sounds of the IPA*, Department of Phonetics and Linguistics, University College London, March 1995; see *http://www.phon.ucl.ac.uk/home/wells/sounds-of-the-IPA.PDF*.
17. L. R. Bahl, P. V. deSouza, P. S. Gopalakrishnan, D. Nahamoo, and M. A. Picheny, "Decision Trees for Phonological Rules in Continuous Speech," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Toronto, 1991, pp. 185–188.
18. N. Mukherjee, N. Rajput, L. V. Subramaniam, and A. Verma, "On Deriving a Phoneme Model for a New Language," *Proceedings of the International Conference on Spoken Language Processing*, Beijing, 2000, pp. 850–852.
19. C. Neti, N. Rajput, and A. Verma, "A Large Vocabulary Continuous Speech Recognition System for Hindi," *Proceedings of the National Conference on Communications*, Mumbai, 2002, pp. 366–370.

Received October 15, 2003; accepted for publication March 24, 2004; Internet publication October 8, 2004

Mohit Kumar *IBM Research Division, IBM India Research Laboratory, Block I, Indian Institute of Technology (IIT), Hauz Khas, New Delhi 110016 (mohitkum@in.ibm.com).* Mr. Kumar is a Technical Staff Member in the speech group in the IBM India Research Laboratory. He received his B.Tech. (Honors) degree in computer science and engineering from the Indian Institute of Technology, Kharagpur, in 2002. Since joining IBM that same year, he has been involved in the area of Indian languages speech recognition and text-to-speech synthesis. Mr. Kumar is currently pursuing a master's degree at the Indian Institute of Technology, New Delhi. His research interests are in speech processing, natural language processing, and bioinformatics.

Nitendra Rajput *IBM Research Division, IBM India Research Laboratory, Block I, Indian Institute of Technology (IIT), Hauz Khas, New Delhi 110016 (rnitendra@in.ibm.com).* Mr. Rajput is a Research Staff Member in the speech group in the IBM India Research Laboratory. He received his M.Tech. degree from the Indian Institute of Technology, Mumbai, in 1998, and his B.E. degree from Government Engineering College, Jabalpur, in 1996. Since joining IBM in 1998, he has been involved in the areas of audiovisual speech recognition, audio-driven facial animation, Indian languages speech recognition, and voice tooling. His research interests are in the fields of statistical signal processing and image processing.

Ashish Verma *IBM Research Division, IBM India Research Laboratory, Block I, Indian Institute of Technology (IIT), Hauz Khas, New Delhi 110016 (vashish@in.ibm.com).* Mr. Verma is a Research Staff Member in the speech group in the IBM India Research Laboratory. He received his M.E. degree from the Indian Institute of Science, Bangalore, in 1997, and his B.E. degree from M. M. M. Engineering College, Gorakhpur, in 1995. Since joining IBM in 1998, he has been involved in the areas of audiovisual speech recognition, audio-driven facial animation, Indian languages speech recognition, and text-to-speech synthesis. He is currently pursuing a Ph.D. degree at the Indian Institute of Technology, New Delhi, in the field of speaker individuality transformation.