INTERNATIONAL JOURNAL OF SPEECH TECHNOLOGY 7, 319–333, 2004

Schwa-Deletion in Hindi Text-to-Speech Synthesis

BHUVANA NARASIMHAN
Max Planck Institute for Psycholinguistics, Nijmegen, The Netherlands
bhuvana@mpi.nl

RICHARD SPROAT
University of Illinois at Urbana-Champaign, USA

GEORGE KIRAZ
AT&T Labs—Research, New Jersey, USA

Abstract. We describe the phenomenon of schwa-deletion in Hindi and how it is handled in the pronunciation component of a multilingual concatenative text-to-speech system. Each of the consonants in written Hindi is associated with an "inherent" schwa vowel which is not represented in the orthography. For instance, the Hindi word pronounced as [namak] ('salt') is represented in the orthography using the consonantal characters for [n], [m], and [k]. Two main factors complicate the issue of schwa pronunciation in Hindi. First, not every schwa following a consonant is pronounced within the word. Second, in multimorphemic words, the presence of a morpheme boundary can block schwa deletion where it might otherwise occur. We propose a model for schwa-deletion which combines a general purpose schwa-deletion rule proposed in the linguistics literature (Ohala, 1983), with additional morphological analysis necessitated by the high frequency of compounds in our database. The system is implemented in the framework of finite-state transducer technology.

Keywords: text analysis, finite-state methods, text-to-speech, phonology

1. Introduction

1.1. The Bell Labs TTS System

In any language, the orthographic representation is often ambiguous and indeterminate with respect to its exact pronunciation (Sproat, 1996; Möbius et al., 1997). For instance, the string $5 is expanded as *five dollars* or as *five dollar* depending on whether it occurs as a prenominal modifier in English (as in *five dollar bill*) (Sproat, 1996). However the written text does not, in itself, provide any clues as to how the string is to be pronounced; additional information about complex noun phrases has to be provided so that the correct linguistic form may be chosen. In Hindi, a similar problem arises with the pronunciation of words containing the schwa vowel which, depending on certain morphophonological factors, is deleted in some contexts but not others. In the Bell Labs Hindi TTS system, one set of linguistic specifications in the text analysis module concerns the pronunciation of words containing the schwa vowel in the orthographic input. In order to produce speech that is intelligible, the written input representation has to be augmented by additional morphological and phonological information so as to accurately predict the contexts in which the schwa vowel in Hindi is deleted. In this paper, we describe the phenomenon of schwa-deletion in Hindi and how it is handled in the pronunciation component of the multilingual concatenative text-to-speech (TTS) system developed at Bell Labs.

We begin with a brief summary of the TTS system as described in Möbius et al. (1996, 1997), Möbius (1999) and Sproat (1996, 1998), before moving on to a detailed discussion of the problem of schwa-deletion in Hindi. The TTS system developed at Bell Labs consists of a set of language-independent modules. The flow of information between the modules is unidirectional—linguistic text analysis followed by the prosodic component which is followed by the synthesis component; a single data structure modulates communication between the modules. A principal characteristic of the system is its multilinguality. Although the software modules are language-independent, the multilingual character of the TTS system derives from information unique to particular languages, such as acoustic inventories and morphophonological rules which are represented in precompiled finite-state transducers and tables, and retrieved at run-time.

In the *text analysis module*, the written form (e.g. a character string from a character set such as ASCII) is mapped onto linguistic parameter specifications which are later converted into parameters (e.g., formant parameters, concatenative unit indices, pitch time/value pairs) that drive the actual synthesis of speech (Sproat, 1996, 1998). Since the written form of a language does not correspond perfectly to the corresponding spoken forms, the text analysis module has to specify a mapping from standard orthographic input to an underlying linguistic representation which provides a rich array of information about tokenization of the input into sentences and words, word pronunciation, accenting, and the assignment of prosodic phrases, among others. The text analysis component thus combines what is traditionally understood by "text normalization" tasks with additional linguistic analysis at the lexical, morphological, and phonological levels.

In addition to tasks such as expansion of abbreviations and prosodic phrasing, multilingual text analysis involves a range of problems which require language-particular analysis. Some examples include the following (see Sproat, 1998 for a fuller discussion of these issues): (a) word boundaries are indicated using whitespace in some languages (German, English, Hindi) but not in others (Chinese, Japanese) where they have to be reconstructed; (b) numeral expansion in English and German differ in the phenomenon of "decade flop"—in German, there is a reversal of the order of decades and units in numbers between 13 and 99; (c) the percentage sign '%' is always read as *percent* in English when denoting a percentage, whereas in Russian, a range of contextual factors have to be taken into account in order to determine its pronunciation (e.g. number, case and gender of the noun following the adjectival form of the word *procent*, among others). The computational framework for text analysis is based on weighted finite state transducers (WFST's), which are constructed using a lexical toolkit which allows for the statement of linguistic generalizations in a readable form. The finite-state framework used in the Bell Labs TTS system is more fully described in Section 3 of this paper.

The *duration module* assigns a duration to each speech sound of the language. Segmental duration is highly context-sensitive, depending on a number of factors which define a relatively large feature space. These features include *phone identity factors* (identity of the current segment, as well as of the previous and next segment(s)), *stress-related factors* (including degree of discourse prominence and lexical stress), and *locational factors* (the location of (a) the segment in the syllable, (b) the syllable in the word, (c) the word in the phrase, and (d) the phrase in the utterance) (van Santen, 1998: 137). Constructing a quantitative model for duration involves using a segmented speech corpus for performing inferential-statistical analysis, and estimating parameters of the model (Möbius and van Santen, 1996). A class of arithmetical models known as "sum-of-products" models can be applied to reliably predict the duration of each segment based on its feature vector (van Santen, 1994).

The *intonation component* takes a representation consisting of phoneme, syllabic stress, phrasing, and accenting information, and computes a fundamental frequency (F_0) contour. The variation in fundamental frequency is a function of a variety of prosodic features ranging from word accentuation to phrase intonation, which has been dealt with by two main classes of intonation models—"superposition models" and "tone sequence models" (van Santen et al., 1998). Superposition models "interpret F_0 contours as complex patterns resulting from the superposition of several components,' whereas tone sequence models generate F_0 contours from "a sequence of phonologically distinct tones, or categorically different pitch accents, that are locally determined and do not interact with each other" (1998: 142). The Bell Labs TTS system uses a superposition style model for languages such as German, French, and Italian, among others—this involves computing an F_0 contour by adding three types of time-dependent curves: a phrase curve (e.g. declarative, interrogative), accent curves (accented syllable followed by zero or more non-accented syllables), and perturbation curves (which take into account the effect of obstruents on pitch in the post-consonantal vowel). A tonal target model is used for Mandarin which involves the derivation of a sequence of tonal targets by applying tonal coarticulation rules to a basic representation of Mandarin tones; effects of sentence intonation and emphasis are then added (Shih and Sproat, 1996).

Acoustic inventory construction involves selecting an appropriate speaker (based on a number of criteria), recording a list of unit types, and selecting the best candidates. The appropriate speech intervals are then excised for storage in the inventory. Specifically, the procedure involves recording diphonic units (units containing the transition between two adjacent phonetic segments) as well as triphones, and selecting optimal units based on criteria such as spectral discrepancy and energy measures. Coarticulatory effects can necessitate the storage of context-sensitive units well. Elements thus selected are stored as tables after normalization. At run time, the necessary units are selected from the inventory, concatenated, and assigned new durations, F_0 contours and amplitude profiles. Parameter vectors are passed on to the synthesis module which uses linear predictive coding (LPC) synthesis together with an explicit voice source model.[1]

1.2. Hindi Text-to-Speech Synthesis

In this paper, we shall be concerned with aspects of the first task in the TTS conversion process for the Hindi language—transforming the input text into a linguistic representation from which synthesis can proceed (Möbius et al., 1996). As mentioned earlier, constructing the text interpretation module for a multilingual TTS system offers particular challenges having to do with the linguistic properties of the language in question. The literature on text analysis for Hindi TTS systems is relatively sparse and we are not aware of any published work which describes issues in Hindi text interpretation for text-to-speech systems in any great detail. Hence we cannot provide an in-depth comparison of our approach to text analysis (and the schwa-deletion problem in particular) to others in the literature. Nevertheless, we will briefly outline several of the main lines of research in this area reported in the literature.

Verma et al. (1995) propose a Hindi text-to-speech synthesis based on syllables as units, along with a special class of 150 consonant clusters to generate unlimited utterances. The combination of the most frequently occuring 29 consonants and 10 vowels (5 long and 5 short) in Hindi give rise to 290 CV and 290 VC syllables. However the possibility of generating short vowels out of corresponding long vowels using durational and frequency rules has reduced the required number of syllables in the database to 290. Text input is fed to the “word parser” which identifies the basic syllables in the word, which are then retrieved from the database and merged to make a parametric file for the given input word. Owing to discontinuities at syllable boundaries, durational and frequency rules are applied to smooth out the discontinuities. Finally, the parametric files is sent to a Klatt synthesizer to generate the sound file.

Rao (1993) describes an integrated speech recognition synthesis system for Hindi (VOICE) which was planned to accept clearly spoken isolated/connected words and produce intelligible speech. Although the study primarily deals with the speech recognition component, the synthesis component is described briefly. A database of 500 Hindi sentences was constructed from a vocabulary of 207 words related to railway reservation enquiries, and 200 of these sentences were labelled into fine acoustic classes. The synthesizer was built based on Klatt’s formant synthesizer (Klatt, 1980), using a synthesis-by-rule approach. The synthesis-by-rule program generates the sequence of appropriate features from a given input phoneme string, and the synthesizer accepts the sequence of features (such as formant frequencies, amplitude, and pitch), and converts it into speech. The synthesizer accepts the standard Hindi phonemes, including aspirated and retroflex consonants, and nasalized vowels, and generates clearly intelligible synthesized speech.

Sen and Samudravijaya (2002) describe a “web reader” which reads out the text in web pages in Hindi or Indian English. The text-to-speech conversion is performed in three stages: text analysis, phoneme to acoustic-phonetic parameter conversion, and parameter-to-speech conversion. The synthesizer uses formant synthesis, based entirely on software, and capable of working on virtually any platform. The *text analyzer* converts text into a phoneme string; the phoneme-to-speech conversion involves mapping specified phonemes into corresponding time-varying acoustic-phonetic parameters using the relevant context-dependent rules, and converting the generated parameters into corresponding speech using a source-filter speech production model (cf. Furtado and Sen, 1996). Sen and Samudravijaya provide a detailed description of the TTS system, however we focus here on their description of the Hindi component of the text analysis system (which is designed to handle input in Hindi as well as English). The textual input is segmented by the main parser and classified into dates, time, currency, alphanumerics, acronyms, abbreviations, special (arithmetical) characters, numbers, names, and the words of the language. Dates, time, and currency are appropriately interpreted; alphanumerics and acronyms are pronounced character by character. Abbreviations and special characters are pronounced on the basis of information provided in an "abbreviation table" and "character table" respectively. A number string is divided into fields, and keywords corresponding to billion, million, thousand etc. are inserted; digits after the decimal point are pronounced character by character. A phonetic dictionary consisting of 8000 words can be accessed by indexing and binary search, and the pronounciation of individual words of the language can be obtained. In the absence of a match, further morphological analysis into prefixes, suffixes, and roots is performed; the pronunciation of the root is obtained from the dictionary and merged with that of prefixes and suffixes following contextual modification as necessary. If this process does not yield a match, the default option is to select the "best" root alternative and use a set of "letter-to-phoneme" rules to obtain a pronunciation for the (multimorphemic) word. Context-dependent rules, including those of schwa-deletion are applied to the text-character string. The pronunciation of various morphs of the word are merged, taking into consideration context-dependent influences of the morphemes, and ultimately phonological rules are applied to the entire phoneme string.

Bhaskararao et al. (1994) describe a demi-syllable based concatenative TTS system for synthesizing spoken Hindi sentences. The different parts of the system are described as follows (1994: 1239–1240). The text formatting component takes as input (Romanized ASCII representations) of the Devanagari script and performs various "preprocessing" operations including the expansion of arithmetic symbols (e.g. the minus symbol $\{-\}$ is converted to {*ghaTA*}) and abbreviations (e.g. {*ku.*} becomes {*kumAri*}). In the next step, the (orthographic) characters are converted into the corresponding phonemes, and morphophonemic rules apply to this representation. These rules include the insertion of /i/ in word-initial consonant clusters such as {*sp*}, {*sp*}, or {*sk*} (e.g. {*stri:*} becomes /istrI/), homorganic nasal conversion, and schwa deletion. The string of phonemes is then converted to segments to which appropriate durations are applied. Pitch variation is achieved by pitch synchronous overlap. The synthesis segment library (with a total of 506 units) consists of demi-syllables as units, along with a selected set of triphones. The selected segments are processed for pitch and amplitude normalization.

A more detailed account of a text analysis component for Hindi text-to-speech synthesis is provided in Bhaskararao and Mathew (1992). The system takes Hindi text input in Devanagari script, and outputs a phonemic transcription of the standard variety, showing syllable division using a lookup lexicon of root forms. The various steps involved in the conversion are text pre-processing, i-epenthesis, *schwa* deletion, anusvAra conversion, miscellaneous changes, and syllable division. The rules of i-epenthesis are text pre-processing are as described in Bhaskararao et al. (1994), which have been summarized above. The process of anusvAra conversion involves converting the anusvAra "*M*" into the phonemes /m/ or /N/, depending on the following homorganic letter, whether it is a bilabial stop or retroflex stop respectively. Elsewhere it is converted to the phoneme /n/. Syllabic division involves the implementation of a syllable division algorithm which groups the output phoneme strings into corresponding syllables. Various miscellaneous changes include specifying exceptions to the normal grapheme-to-phoneme correspondences. Thus, for instance, the sequence "*jn*" is converted to the phonemic sequence /gy/, and conversely the regular velar fricative grapheme "*h*" and the *visarga* grapheme "H" are mapped into the phoneme /h/. Finally, the problem of schwa-deletion is dealt with by specifying a range of conditions in which the schwa is deleted:

(a) If the input word matches a word in the lexicon which is followed by the schwa vowel, then the word-final schwa is deleted.
(b) If the word ends in a consonant cluster, then the schwa is retained.
(c) When the schwa-vowel does not belong in the first syllable of the verb and is followed by a range of suffixes such as "*vAnA*", "*nA*", "*nI*", "*ne*", then it deletes (thus "*calavAnA*" becomes "*calvAnA*").
(d) When a word can be diagnosed as having a prefix such as "*dur*", "*an*", "*nir*", etc., the prefix is chopped and the residue is subjected to the schwa-deletion process (thus "*durupayoga*" becomes "*durupyog*").
(e) If the second syllable of a word contains a schwa, and it is preceded by a consonant (or a cluster consisting of a nasal phoneme and consonant) and followed by a consonant, the schwa is deleted (e.g. "*ahamad*" becomes/ahmad/).

(f) If the result of a schwa-deletion operation is two separate words, then the rules of schwa deletion are applied to each of these words. Thus, the schwa vowel following the first word in the compound "*ahamadanagara*" is deleted to give "*ahamad*" and "*nagar*", and then schwa deletion applies to each of these words to give the final output /ahmadnagar/.

From this brief survey, we can conclude that different approaches to the problem of building TTS systems for Hindi have been attempted, including formant-based synthesis, demi-syllable-based concatenative synthesis, and syllable-based synthesis, all of which contrast in interesting ways from the diphone-based concatenative synthesis system using LPC that is used in the Bell Labs Hindi TTS system. Of these, only Sen and Samudravijaya (2002) and Bhaskararao and Mathew (1992) discuss the text analysis component in any detail, and only the latter provide the specifics of dealing with the schwa-deletion problem in Hindi text-to-speech synthesis. The main characteristic of the approach advocated in Bhaskararao and Mathew (1992) is the use of a list of specific environments for schwa-deletion and the availability of an online lexicon of words in the language for lookup purposes. As will become evident from the discussion in the remainder of the paper, our approach differs in that we capture the generalization inherent in the different environments for schwa-deletion with a general rule of schwa-deletion in word-final contexts as well as a (context-sensitive) schwa-deletion rule for word-internal contexts, which is based on the linguistic analysis provided in Ohala (1983). The rule applies to any string delimited by whitespace (corresponding approximately to words) and does not involve a process of matching the input word with actual words in the language as specified in a lexicon. Further, our rule (discussed more extensively in Section 3.3) differs from condition (e) proposed above in requiring the presence of *both* a consonant and a vowel to the right of the schwa in order for it to delete, rather than just a single consonant (which wrongly predicts that a word such as "kalam" 'pen' would become "kalm"). Other than these differences, there are commonalities in the two approaches in that our system also make use of lists of prefixes and compounds which provide a level of morphological analysis for the word before it is inputted to the schwa-deletion rule. In the following sections, we shall proceed to discuss, in considerable detail, the different aspects of our solution to the problem of schwa-deletion in Hindi TTS.

2. Schwa-Deletion in Hindi Speech Synthesis

We begin by providing some background information about the Hindi language relevant to understanding how we address the problem of schwa-deletion. Hindi is an Indo-European language derived from Sanskrit, and is spoken by more than 350 million people around the world. The language has 10 oral vowels, each of which has a nasal counterpart, and 30 consonants. An additional 5 consonants and two vowels which occur mainly in (Persian, Arabic, and English) loan words can also be considered to be part of the phoneme inventory of Hindi owing to the frequency with which the loanwords are used in the language (cf. Ohala, 1999).[2]

Hindi belongs to the Indo-Iranian branch of the Indo-European family of languages. Two distinctive features of its phonology include aspiration and retroflexion in its consonant inventory; stress is not distinctive in the language, and is tied to syllable weight (Kachru, 1987). Most derivational and inflectional morphology in Hindi is affixal; forms of nouns undergo changes to indicate number, gender, and case, and prenominal adjectives agree with the head noun in number, gender, and case (where case relations are indicated by postpositions) (1987: 477–479). Verbs express various aspect and mood distinctions, while tense distinctions are expressed by forms of an auxiliary copular verb (p. 480–481). Verbs occur in morphologically related sets (causal verbs) or in compounds, where the second "explicator" verb adds to or restricts the meaning of the main verb (p. 483).

The Hindi language is written from left to right using the Devanagari script, which is a slightly modified version of the one commonly used in Sanskrit (McGregor, 1995). Each of the consonants in written Hindi is associated with an "inherent" schwa vowel which is not represented in the orthography (see Table 3 for the list of phonemes in the Hindi language). For instance, the Hindi word "namak" pronounced as [nəmək] ('salt') is represented in the orthography using the consonantal characters for [n], [m], and [k]. The intervening schwa vowels are associated with each consonant by the speaker while pronouncing the written word. Vowels other than the schwa (e.g. the vowels [ɪ] and [ʊ] in the word pronounced as [ʃɪʃʊ] 'child') are represented overtly in the orthography with a variety of diacritic and non-diacritic markers surrounding the consonant.

In a multilingual TTS system, the absence of an overt glyph representing the "inherent" schwa vowel in the orthographic input is handled by a general rule which inserts a schwa in the lexical string between consonants in the text-analysis component of the system. However, a number of facts complicate the issue of schwa pronunciation in Hindi (Ohala, 1983). First, as might be noticed in the above example, not every schwa following a consonant is pronounced. That is, the schwa associated with the word-final consonant [k] in the written form of the word pronounced as **[nəmək]** is deleted. In fact, every schwa in the final position of orthographic words is elided in Hindi. Further, even within the (multimorphemic) word, the schwa can be deleted when it appears in certain positions. For instance, the schwa following [m] in the word "namak[ii]n" ('salty') (pronounced **[nəməkin]**) is usually deleted to produce the word pronounced as **[nəmkin]**.[3] Finally, the presence of a morpheme boundary can block schwa deletion where it might otherwise occur. The schwa preceding [g] in "ajagar" ('python') is deleted to produce "ajgar", but the schwa preceding [g] in the word "mah[aa][++]nagar" ('great city') cannot delete to produce "*mah[aa][++]ngar", owing to the morpheme boundary (represented as [++]) following the prefix "mah[aa]".[4]

Clearly, a more complex account is required to predict the contexts in which the Hindi schwa is pronounced in order to produce intelligible speech. While a number of solutions for the schwa deletion problem has been proposed in the linguistics literature (Ohala, 1983; Pandey, 1989), we rely principally on Ohala's account of the phenomenon. In the following discussion, we describe how we deal with the problem of schwa pronunciation in Hindi, using ordered rules supplemented by morpheme boundary information provided in the form of lists.

3. Architecture

3.1. Overall Architecture

The schwa-deletion system is based on a finite-state framework which makes use of weighted finite-state transducers constructed using **Lextools**, a lexical toolkit for the descriptions of lexica, morphological rules and phonological rules, and morphosyntax, *inter alia* (Sproat, 1997; see also Kiraz and Möbius, 1998; and see http://www.research.att.com/sw.tools/lextools for a downloadable version of the toolkit). A set of transducers (FST's) compute the mappings between several different levels of linguistic descriptions: between the surface orthographic level and a lexical level, and between the lexical level and the level of phonological representation. In what follows, we assume some familiarity with finite-state acceptors (FSA's), regular languages, and regular expressions (Hopcroft and Ullman, 1979).

FST's are similar to FSA's; they are computational devices consisting of a finite number of states, a designated start state, a set of designated final states, and a finite set of directed labeled arcs between states. Arc labels of FST's consist of pairs (more generally n-tuples) of symbols from a finite alphabet. The interpretation of a label $x{:}y$ on an arc between states $s1$ and $s2$ is as follows: if the machine is in state $s1$ and the next token of input is x, then the machine may consume the x, output a y and proceed to state $s2$. If the machine is in a state where there is no arc leaving it which is labeled with an input symbol corresponding to the next token of input, then the machine cannot proceed any further from that state. A computation using FST T on input w is successful just in case one can start in the initial state of T, at the left edge of w, and consume symbols of w, arriving in a final state of T, outputting a string y, where y is determined by the output labels on the arcs traversed during the computation.

Algebraically, FST's compute regular relations. Regular relations are (possibly infinite) sets of pairs (more generally n-tuples) of strings that can be constructed using one or more of the following operations: concatenation, transitive closure (Kleene star), and union. Regular relations are said to be closed under these operations: that is if A is a regular relation and B is a regular relation, then so is AB, where AB consists of each element of A concatenated with each element of B (the cross-product); similarly, if A is a regular relation, then so is A*, where A* represents the n-way concatenation of any member of A with any member of A, for n from 0 to infinity; finally, if A and B are regular relations then so is $A \cup B$.

Regular relations (and FST's) also closed under composition and inversion. Given two relations R1 and R2, the composition of R1 and R2, denoted $R1 \circ R2$ is a regular relation that relates strings in the domain of R1 with strings in the range of R2; see (Mohri, 1997; Mohri et al., 1998) for a description of algorithms for computing composition of FST's. The inversion of a relation R, R^{-1}, simply swaps the domain and range of R; to invert an FST, one merely needs to swap the input and output labels on each arc. These closure properties are particularly useful in natural language applications of FST's. Closure under composition means that one can develop simple rules, implemented as FST's, and compose them together to produce a single FST that implements the entire cascade of rules; this provides for a straightforward implementation of ordered rewrite rules (Kaplan and Kay, 1994). Closure under inversion means that one can create a system that is generative—e.g. one that maps from Hindi phonological representations into Devanagari text, and then invert the resulting transducer to produce a system that computes the relation in the other direction.

Three other points need to be made. First, either the input or the output label may be the empty string, denoted conventionally as ε. If the input label on an arc is ε, and the output symbol is a non-empty symbol x, denoted $\varepsilon{:}x$ then the machine performs an insertion, consuming no input, but inserting an x. Similarly, if the label is $x{:}\varepsilon$, then the machine performs a deletion, reading an x and outputting nothing. Secondly, FST's may be (and often are) non-deterministic, in that they may produce more than one output for any given input; in this case the output can be represented as a lattice, which is simply an FST. This can be useful if the task is to produce a set of plausible alternatives, which can be further disambiguated by other processes. For example, an input word may be ambiguous between several different morphological analyses; one's lexical transducer can produce a set of possible outputs for this word, which could be disambiguated using further FST's representing syntactic constraints. Finally, one can also add weights to the arcs, where weights are (usually) real numbers that might, for example, represent the probability of a particular transition in the machine, or they might simply represent some hand-constructed weight. Selection of alternative analyses given a set of weighted paths is accomplished using a shortest (cheapest) path or "best-path" algorithm (Mohri, 1997; Mohri et al., 1998). There are various interpretations of how the weights are combined along a path (are they summed or multiplied?), and what counts as cheapest (higher or lower value). In what follows we will always assume that weights are summed along a path through a lattice, and that "cheaper" means having lower weight.

The rules and entries used as input to the lextools suite of tools are stated in terms of an extended regular expression language. Standard symbols are ε to represent the empty string, and Σ to represent the alphabet of symbols; ":" is used to represent the mapping between an input and output label as above; finally weights are represented by numbers within angle brackets: thus, $\langle 0.5\rangle$. One of the tools in the lextools suite implements the weighted rewrite rule algorithm of (Mohri and Sproat, 1996), which followed on earlier work in Kaplan and Kay (1994). This allows one to write rewrite rules that look very much like the rewrite rules familiar to linguists. Lextools thus has much in common with other finite-state toolkits, such as the Xerox toolkit (see http://www.xrce.xerox.com/competencies/content-analysis/fsCompiler/), though it differs from these in allowing weights.

Our discussion in Section 2 collapsed two phenomena which are usually distinguished for the purposes of linguistic analysis. The first is the problem of pronouncing the "inherent" schwa associated with consonants and its ellipsis at the end of words in the orthography. The second phenomenon has to do with the deletion of the schwa in one of the derived phonological representations of the word. We treat both aspects in the following discussion.

3.2. Orthographic Schwa Rule

The first component of the schwa-pronunciation module inserts a schwa vowel after consonants and elides it word-finally. We implement this in the "reverse direction" by starting with a level of lexical representation in which the schwa occurs after every consonant. A general rule then deletes the schwa globally:[5]

$$a \rightarrow \varepsilon / \$\,[\mathrm{Orth}]?\mathrm{C}+\ \_\_;$$

where C+ stands for consonant sequences consisting of one or more consonants (the '+' symbol is often called "Kleene plus", and represents one or more catenations of a set with itself, rather than the zero or more catenations denoted by Kleene star), and $ for a syllable boundary. The symbol [Orth] stands for orthographic symbols which might be (optionally) present before the consonant, and the '__' represents the locus for the substitution of the schwa vowel with ε (following standard conventions in linguistics). The operator '?' is for optionality. Thus this rule deletes an /a/ that occurs after a syllable boundary, an optional orthographic symbol and one or more consonants. The rule can be compiled into a transducer (again, using the algorithm of Mohri and Sproat, 1996), and then inverted to produce a transducer that takes as input the surface orthographic representation (without postconsonantal schwa vowels), and returns a lattice of possible lexical outputs.

3.3. Word-Internal Schwa-Deletion

Having implemented a general rule of postconsonantal schwa insertion with word-final schwa-deletion in the orthographic input, we now constrain the contexts of the application of the rule word-internally on the basis of the rule suggested in Ohala (1983). This is done with a combination of further rules and filters.

We begin with a (simplified version of the) rule which we use to replace the schwa with an intermediate symbol [NullV] at *morpheme boundaries*:

Schwa-replacement rule 1:

$$a \rightarrow \ ([\mathrm{Null}V]|\mathrm{a}\langle 1.0\rangle)/[++](\Sigma - [++])^*(\mathrm{V}) \times (\Sigma - [++])^* \_\_ [++];$$

Here V stands for the class of vowels, and the operators "−" is subtraction, and the string "(Σ − [++])" stands for any character which is not a morpheme boundary. The rule applies right-to-left. The schwa vowel goes to [NullV] or is retained, but at a cost of ⟨1.0⟩. The string "Σ − [++]" stands for any character which is not a morpheme boundary. Hence, when the input consists of a word with internal morpheme boundaries, e.g. "saha[++]karmii", the schwa-replacement rule (1) (correctly) replaces the schwa following the prefix-final schwa with a null vowel symbol to produce "sah[NullV][++]karmii". Similarly, words such as "para[++]lok" ('the next world') or "para[++]janm" ('subsequent birth) become "par[NullV][++]lok" and "[par[NullV][++]janm" respectively.

A second rule maps the schwa vowel to the intermediate symbol [NullV] *within* the morpheme. Ohala (1983: 139–140) suggests a general purpose rule which would delete postconsonantal schwa vowels in the correct contexts. A (slightly modified) version of her rule maps the schwa vowel to the intermediate symbol [NullV] (which is ultimately deleted):

Schwa-replacement rule 2:

$$a \rightarrow ([\mathrm{NullV}]|\mathrm{a}\langle 1.0\rangle)/\mathrm{VC(C)}\ ?\_\_[++]?\ \mathrm{C}[++]?\ \mathrm{V};$$

Here, V stands for the class of vowels, C stands for the class of consonants, and the possibility of an optional consonant following C stands for consonant clusters. As mentioned before, [++] represents a morpheme boundary. The rule is marked to apply from right to left (Kaplan and Kay, 1994). Since schwa-deletion is optional, the rule allows two alternative outputs; one which replaces the schwa with the [NullV], the other which retains the schwa, at a cost. All other things being equal, the one with the retained schwa will be disfavored since it will involve a more expensive analysis. The application of the rule is limited by two constraints. Any morpheme boundary to the immediate left of the context for schwa-deletion blocks the application of the rule. The left context specifies that a consonant (cluster) must precede the schwa for it to delete. Since a morpheme boundary to the immediate left of the consonant (cluster) would block schwa deletion, no optional morpheme boundaries are specified in the left context of the rule. The right context of this rule specifies that schwa deletion only occurs before a consonant followed by a (non-schwa) vowel. Optional morpheme boundaries are allowed to occur to the right of the schwa (since these do not interfere with the application of the rule). The schwa-deletion rule is further constrained by a filter (to be described further in Section 3.5) which ensures that any consonant cluster created as a result of schwa deletion must satisfy the phonotactic constraints on consonant clusters in Hindi.

This rule works in the following way. Within monomorphemic words, the schwa (indicated in bold font in the examples below), is (replaced by the intermediary symbol [NullV] which is ultimately) deleted when the conditions specified by the rule are met:[6]

m[a a]**la** t[ii] →	m[a a]l t[i i]	(female proper name)
V C**a**C V	V CC V	
m a z**a** b[uu]t →	m a z b[u u] t	('strong')
VC**a**C V →	VCC V	
u l **a**[j h]an →	u l [jh]a n	('problem, complicaton')
VC**a** C V	VC C V	
j a n g**a**l[ii] →	j a n gl[i i]	('pertaining to the forest,
VCC**a**C V	VCCCV	wild')(Ohala, 1983:128)

When a suffix (e.g. "-[ii]n" or "-[au]t[aa]") is added to a word such that the new word satisfies the input condition to the schwa-deletion rule, the schwa is duly deleted (note that morpheme boundaries may freely occur in the right context of the rule). As mentioned above, we indicate a morpheme boundary by '[++]':

```
n a m a k[++][i i]n
 V Ca C       V
   → n a m k[++][i i]n          ('salty')
        VCC       V
s a m a[jh][++][au]t[aa]
 V Ca   C   V
   → s a m [j h][++][a u]t[a a]    ('understanding')
      VC C          V
```

When a prefix is added to a word such that a morpheme boundary appears to the left of a schwa-deletion context in a word, the schwa-deletion rule fails to apply, as expected:

```
     p r a[++]ga t i → p r a[++]g a t i   ('progress')
       V     CaCV        V    C aCV
anu[++]kara[n.] → anu[++]kara[n.]('imitation')
     V      CaCV          V       CaCV
```

3.4. *The Prefix Lexicon*

Since the schwa-replacement rule is sensitive to the internal morphological structure of the word (morpheme boundaries in the left context of the rule), we begin by creating a lexical database with prefix information derived from online corpora and augmented with lists provided in Singh and Agnihotri (1997). A morpheme boundary (represented by [++]) was introduced within any word beginning with character strings corresponding to Hindi prefixes such as "antar-", "anu-", "para-" or "ku-". We stipulated that morpheme boundary insertion could only occur in words with at least three characters (each character represented by the symbol Σ) following the prefix. This avoided the erroneous insertion of internal morpheme boundaries in monomorphemic words beginning with the same characters as those in our list of prefixes. Each character in the word is given an arbitrary cost of $\langle 0.5 \rangle$, and the length of the word determines the cost associated with the word as a whole (since costs are additive). The final line of the word list associates a default cost to any character string. A partial list is given below:

antar (ε : [++])($\Sigma\langle 0.5\rangle$)($\Sigma\langle 0.5\rangle$)($\Sigma\langle 0.5\rangle$)
  + ('inner, interior')

anu(ε : [++])($\Sigma\langle 0.5\rangle$)($\Sigma\langle 0.5\rangle$)($\Sigma\langle 0.5\rangle$)
  + ('after, according to')

para(ε : [++])($\Sigma\langle 0.5\rangle$)($\Sigma\langle 0.5\rangle$)($\Sigma\langle 0.5\rangle$)
  + ('other, distant, subsequent')

pari(ε : [++])($\Sigma\langle 0.5\rangle$)($\Sigma\langle 0.5\rangle$)($\Sigma\langle 0.5\rangle$)
  + ('around')

ku(ε : [++])($\Sigma\langle 0.5\rangle$)($\Sigma\langle 0.5\rangle$)($\Sigma\langle 0.5\rangle$)
  + ('bad, defective')

saha(ε : [++])($\Sigma\langle 0.5\rangle$)($\Sigma\langle 0.5\rangle$)($\Sigma\langle 0.5\rangle$)
  + ('together, along with')

($\Sigma\langle 0.5\rangle$)+

The notation 'ε:[++]' implements the insertion of a morpheme boundary between the prefix and the stem. The final line "($\Sigma\langle 0.5\rangle$)+" represents the case where a word is not morphologically analyzed into a prefix-plus-suffix combination. A list of such regular expressions can then be compiled into a finite state transducer that represents a schematic lexicon for Hindi. An input string such as "sahakarm[ii]" ('fellow-worker', 'colleague') can then be composed with this acceptor. This will yield one or more possible analyses; for "sahakarm[ii]" (with no morphological analysis), and "saha[++]karm[ii]", analyzing the "saha-" as a prefix. The shortest-path algorithm will favor the latter since in the former case every symbol will get a cost of 0.5 (for a total cost of 4.5) whereas in the latter case, the "saha-" prefix comes for free and the total cost (2.5) is cheaper. Similarly, input strings such as "anujanm[aa]" ('have a following birth'), "kukarm[ii]" ('villanous'), "pari[bh][aa][sh][aa]" yield "anu[++]janm[aa]", "ku[++]karm[ii]", and "pari[++][bh][aa][sh][aa]" respectively.

3.5. *Consonant Cluster Filter*

The outputs of these rules have to be filtered, to disallow words which violate consonant cluster constraints in Hindi. For instance, in words such as "sargamE" ('melodies'), the schwa-vowel following the "g" does not delete, since the resultant consonant cluster "rgm" is not attested in Hindi. Similarly, schwa-deletion cannot take place in words such as "adrakE" ('ginger roots') or "shalgamE" ('turnips') to produce "adrkE" and "shalgmE" since they produce illegal consonant clusters such as "drk" and "lgm", respectively.

We derived a list of consonant clusters illegal in Hindi from Ohala (1983), but modified it to exclude clusters such as "r[th]n", "mt", "[rth]", "jn", and "hr" since words with these clusters are attested in the language, albeit rarely, (e.g., "pr[aa]r[th]n[aa]" 'prayer', "imtih[aa]n" 'examination', "saar[th]ak" 'significant', "ajnab[ii]" 'stranger', and "dohr[aa]n[aa]" 'repeat'). The set of strings which are impossible is expressible by a regular expression, given partially as follows::

Consonant cluster filter:

$$!(\Sigma * ((\mathrm{rw[NullV][t.]})|(\mathrm{rk[NullV][t.]})|\ldots| \\ (\mathrm{b[NullV]t}))\Sigma *)$$

The "..." represents other illegal clusters that we omit here for lack of space. The exclamation mark '!' represents complementation (regular languages, though not in general regular relations, are closed under complementation), so that the expression disallows any string containing one of the specified sequences, such as "rw[NullV][t.]"; more specifically, "Σ* rw[NullV][t.] Σ*", will match any string that contains the sequence " rw[NullV][t.]", and " ! (Σ* rw[NullV][t.] Σ*)" will disallow any such string. Thus, although strings such as "karw**a**[t.]E" ('sides'), "kark**a**[t.][ii]" ('female crab'), and "kudr**a**t[ii]" ('natural') constitute instances where schwa-deletion might be expected to takes place, the outputs of schwa-deletion—"karw [t.]E" ('sides'), "kark [t.][ii]" ('female crab'), and "kudrt[ii]" ('natural') respectively—contain consonant clusters which are ruled out by the phonotactics of the language. The consonant cluster filter inserts the [NullV] symbol in place of the schwa in such strings. Since any word which undergoes the schwa- replacement rule will have the intermediary [NullV] symbol in it, any word containing a string which has a [NullV] intervening between the illegal consonant clusters will be ruled out as impossible. The (higher cost) alternative output of our schwa-replacement rule (2) above, in which schwa is retained, is then selected as the only alternative. Finally we delete the intermediary symbol [NullV] by mapping it to the null symbol via a "schwa-deletion FST".

3.6. *Ordering the Components of the Schwa-Deletion System*

Since postconsonantal schwa-insertion and word-final schwa-deletion occur in the initial mapping from the orthographic string to the lexical level of representation, the "orthographic schwa FST" is ordered before the others. The internal morphological structure of the word has to be represented in the input to the schwa-replacement rule, hence the prefix FST is ordered next. The output is a lattice of lexical analyses of the input string each of which is associated with a particular cost. The best-path is selected, using a shortest path algorithm (Mohri et al., 1998). The schwa-replacement FSTs, the consonant cluster FST, and the schwa-deletion FST are then composed together (in that order), and the resulting "phonological schwa FST" is applied to the best lexical analysis outputted by the prefix FST.

For example, let us consider the word "namak[ii]n" ('salty'), which, in the orthographic representation, would lack the schwa vowels, as in "nmk[ii]n". The orthographic schwa FST will output "namak[ii]na" as well as "namak[ii]n"; and the best-path algorithm will select the latter, since it requires fewer number of insertions, and is hence weighted lower. This output ("namak[ii]n") passes through the finite-state lexicon of prefixes, but does not receive any internal morphological boundaries since none of the prefixes in the list matches the initial characters in the input string. As a result, when the phonological schwa FST is applied to "namak[ii]n", the second schwa-vowel is replaced by null vowel since it meets the criteria for deletion as specified in the schwa replacement rule (2) to produce "nam[NullV]k[ii]n". The unmodified string "namak[ii]n" is also outputted, however a higher cost is associated with it. Both outputs successfully pass through the consonant cluster filter, since the cluster "m[NullV]k", created as a result of the schwa-replacement rule is not an invalid string as specified in the consonant cluster list. The best-path is selected, in this case, the string which has correctly undergone schwa-deletion, viz. "namk[ii]n".

4. Further Elaborations in the System: Compounds

While the above system works well with individual words which are separated by whitespace in the written text, longer compounds comprising names and common nouns are problematic, since the morpheme boundary between the members of the compound is not represented by whitespace. As a result schwa-deletion incorrectly fails to take place word-finally when the word occurs as the non-final member of a compound (since the boundary between the members of the compound is not indicated by white-space). Further, schwa-deletion incorrectly occurs in some (compound) word-internal contexts, since morpheme boundary information is not indicated within the compound.

As an instance of these two types of errors, let us consider the compound "loksa[bh][aa]" which consists of the nouns "lok" ('people') and "sa[bh][aa]" ('assembly'). The form of the compound prior to the application of the schwa-pronunciation rules is "loks[bh][aa]". When this string is inputted to the orthographic schwa transducer, the output is "lokasa[bh][aa]" and not the correct form, "loksa[bh][aa]". The absence of a white-space after the word "lok" causes it to be treated as part of a word, not a separate word, hence the orthographic schwa FST does not output the form "lok". When the string "lokasa[bh][aa]" next passes to the phonological transducer, the second type of error occurs. Since the schwa following the letter "s" in the second member of the compound "sa[bh][aa]" meets the condition for the schwa-replacement rule, it is (incorrectly) replaced by the intermediate [NullV] symbol and subsequently deleted. However, if the compound were treated as a simple word with internal morphological structure (just as prefixed words are), then both types of errors would be avoided. That is, if the morpheme boundary were correctly marked as in "loka[++]sa[bh][aa]", the schwa-replacement rule (1) would correctly delete the schwa following the "k" (owing to the presence of a morpheme boundary to the right). Also, the schwa-replacement rule (2) would fail to apply owing to the presence of a morpheme boundary to its left, and the schwa following the "s" in the second morpheme of the compound would not be incorrectly deleted.

In order to implement this, we replicated the method used for representing prefix information, discussed above. We created finite-state lexicons which inserted morphological boundaries between the members of compounds for both proper names and "common noun" compounds. An online database of Hindi newspaper texts consisting of over 5 million words was used to create these databases. The most frequently occurring words of 8 characters or more were retrieved. Monomorphemic strings were discarded, and the list was hand-annotated for morphological boundaries. The combined databases consisted of 1646 entries. A partial list is given below for illustrative purposes:

dila(ε : [++])casp	('interesting, attractive')
catura(ε : [++])[bh]uj	('four-armed')
k[s.]ati(ε : [++])grast	('damaged')
gati(ε : [++])vi[dh]i	('manner of acting, procedure')
candra(ε : [++])[sh]e[kh]ar	(male proper name)
uttara(ε : [++])prade[sh]	(name of an Indian state)
jaga(ε : [++])mohan	(male proper name)

As before, the symbol (ε:[++]) represents the insertion of a morpheme boundary [++] between the two members of a compound (e.g. "dila" and "casp" in the first line of the list above).[7] This lexicon is then compiled into an FST and unioned with the prefix FST given before. The compounding thus applies prior to the schwa-deletion component.

5. Evaluation

The schwa-deletion module was evaluated using textual materials derived from the online corpus of newspaper texts mentioned earlier. We isolated a subset of 13,500 long sentences from the corpus, and ran the TTS system with the schwa-deletion module on this set. We then selected the first 15 words occurring in every block of 500 sentences where schwa-deletion (should have) applied. These were then uniquely sorted to filter out exact repetitions, yielding a database of 290 words. Pronunciation performance was evaluated by manually checking the correctness of the transcriptions.

A transcription was considered correct if the schwa was deleted in a context where schwa-deletion was expected to occur. Multiple (correct) deletions within the same word were treated as a single correct case. A transcription was considered incorrect if (a) the schwa was not deleted in the appropriate context, or if (b) the schwa was deleted in an inappropriate context (where it should have been retained). Multiple mistakes within the same word were considered to be a single error.[8] Table 1 summarizes the results.

The schwa-deletion system performs correctly 89% of the time. An analysis of the errors shows that the schwa is incorrectly retained in contexts where it should

Table 1. Performance the schwa-deletion system.

Total number of words	290
Total correct	258/290 (89%)
Total errors	32/290 (11.3%)
Schwa incorrectly retained	13/32 (40.6%)
Schwa incorrectly deleted	12/32 (37.5%)
Incorrect retention & deletion (in the same word)	6/32 (18.8%)
Unknown	1/32 (3.1%)

have been deleted for 40.6% of the cases. For instance, the schwa following the "p" in "nip**a**[t.][aa]n[aa]" ('to complete') is erroneously retained even though it occurs in a context which meets the conditions of the schwa-replacement rule number 2 (Section 3.5). The schwa is incorrectly deleted 37.5% of the time (e.g., the schwa preceding "s" in "paraspar" ('mutual, reciprocal') is inappropriately deleted). Only in a few cases do both types of errors occur. Thus, given the compound "ar[th]**a**vyavas[th][aa]" ('economy management'), the correct output would be "ar[th]vyavas[th][aa]". The schwa following "th" should be deleted since it occurs at the end of the first morpheme of the compound ("ar[th]**a**"), but the schwa following "y" in the second member of the compound ("vyavas[th][aa]") is retained since there is a morpheme boundary to its immediate left. However, since the schwa-deletion module failed to insert a morpheme boundary between the members of the compound, the system returned the equivalent of "ar[th]avyvas[th][aa]" (where the schwa following "[th]" is incorrectly retained and the schwa following "y" is incorrectly deleted). In the single remaining case in Table 1 where the error type is glossed "unknown", there was some ambiguity in judgement as to whether the schwa should have been deleted or retained. This case consisted of the word "mah[aa]vidy[aa]l**a**y[ii]n" ('pertaining to college') where it is not clear whether the schwa preceding the [y] should be deleted or not.

Table 2 shows the error breakdown on the basis of the source of the error. That is, the schwa-deletion errors in Table 1 were produced owing to incomplete information in the consonant cluster lexicon, the compound boundary lexicon, or the prefix lexicon. In almost 60% of the cases, the schwa-deletion module produced an incorrect analysis because the prefix boundary had been wrongly inserted in a monomorphemic word (Section 3.4). Thus, the word "nipa[t.][aa]n[aa]" is mistakenly assumed to consist of a prefix "ni" attached to the word "pat.aanaa" thus incorrectly blocking the deletion of the schwa following the "p" (owing the presence of a morpheme boundary to its left). In 28.1% of the cases, a morpheme boundary had not been inserted because the compound was not listed in the list of compounds (Section 4). The consonant cluster list which filtered out the outputs of the schwa-deletion rule with improbable consonant clusters proved to be inadequate for 9.4% of the cases (Section 3.5). For example, the deletion of the schwa following the first word "hasta" in the compound "hastaks.ep" would create the consonant cluster "stks." which does not occur in Hindi. However, the schwa-deletion module incorrectly deletes it since the consonant cluster filter list does not list this cluster as being ill-formed. The single case listed as "unknown" is identical to the ambiguous word in Table 1.

Table 2. Breakdown of error types.

Consonant cluster	3/32 (9.4%)
Compound boundary	9/32 (28.1%)
Prefix boundary	19/32 (59.4%)
Unknown	1/32 (3.1%)

6. Areas for Future Work

While the relatively low error rate of the schwa-deletion system shows that it performs well in a majority of the cases, the error breakdown suggests a number of areas of further improvement. The list of consonant clusters needs to be augmented since it fails to filter out obviously impossible clusters in Hindi such as "stks." While lists of possible and impossible consonant clusters in Hindi can be obtained from grammars or linguistic studies, they can be augmented by information derived from online corpora to derive frequencies of occurrence of within-word consonant clusters in the language. The probabilistic information can be represented in the form of weights associated with each consonant cluster in a list. Rather than ruling out a consonant cluster altogether (as we did in the list discussed in Section 3.5), a word containing a particular cluster (formed as a result of schwa-deletion) would be associated with a particular weight derived from its frequency of occurrence in the corpus. A string of consonants associated with a high weight would be extremely rare or absent in the corpus, whereas a low weight would indicate relatively high frequency of occurrence. The output would be a lattice of words associated with different weights, and

Table 3. Table of phonemes (adapted from Ohala, 1983, 1999).

		Consonants								
		Bilabial	dental and alveolar	velar	retroflex	palato-alveolar	palatal	glottal	uvular	labio-dental
Plosive	voiceless unaspirated	p	t	k	[t.]				q	
	voiceless aspirated	[ph]	[th]	[kh]	[t.h]				x	
	voiced	b	d	g	[d.]				G	
	breathy-voiced	[bh]	[dh]	[gh]	[d.h]					
Affricate	voiceless unaspirated					c				
	voiceless aspirated					[ch]				
	voiced					j				
	breathy-voiced					[jh]				
Fricative	voiceless		s		[s.]		[sh]	h		f
	voiced		z							
Flaps	voiced				[r.]					
	breathy-voiced				[r.h]					
Nasals		m	n	[ng]	[n.]	[n∼]				
Approximant							y			v
Taps			r							
Laterals			l							

Vowels

	Front		**Central**		**Back**	
	oral	**nasal**	**oral**	**nasal**	**oral**	**nasal**
High	[ii]	[II]			[uu]	[UU]
	i	SI			u	U
Mid	[e]	E	a, H	A	o	O
	[ai], %*	[AI]			[au], @*	[AU]
Low	Y					
			[aa]	[AA]		

*% and @ correspond to the vowels in English borrowings such as the vowel [o] in “box” and the vowel [oe] “map” respectively.

the best path algorithm would select the output with the lowest weight.

Further improvement in the performance of the system can be obtained by modifying the prefix list. The mis-segmentation of ambiguous strings (e.g. the word “nipa[t.][aa]n[aa]” could be monomorphemic or contain a prefix boundary following “ni”) can be reduced by stipulating that a prefix boundary can only be inserted when followed by at least five characters (instead of three characters in the current version). This would reduce the number of instances where the prefix boundary is gratuitously inserted in shorter words. A better solution would be to build an online lexicon of Hindi (which is currently unavailable) so that we have more complete information about which words are monomorphemic.

In order to increase coverage of compounds, a compositional model of compounds can be created. The tool allows one to write a finite-state grammar describing words of arbitrary morphological complexity and length (Sproat, 1995). Based on linguistic knowledge of constraints on which words can be non-final and which the final members of a compound, arbitrarily long concatenations of names and common nouns can be decomposed (see Jannedy and Möbius (1997)) for a name-pronunciation system using weighted FSTs).

The prefix information discussed above could be incorporated into such a morphological analysis FST as well.

One further possibility for future exploration involves alternative models of schwa-deletion in the linguistic literature. For instance, the deletion of schwa has been predicted to occur in unstressed syllables in certain contexts (Pandey, 1989). While the role of internal morphological complexity has not been sufficiently analysed in such models, further investigation could reveal a more elegant solution using general principles rather than item-specific lists which would be both more economical and more accurate.

Notes

1. Non-source-filter approaches such as *pitch-synchronous overlap and add* (PSOLA) which are based on waveform concatenation (Charpentier and Moulines, 1990), in principle, allow for systems with higher speech quality, although high sensitivity to the precise cut-point location of waveform segments and handling extreme variations in pitch remain problematic for waveform concatenation (Shih and Sproat, 1996). One possible solution is to reduce the number of concatenations (as well as the amount of signal processing required to correct prosodic properties of the units) by selecting the longest strings of phonetic segments from a large speech database (Möbius, 2000). The selection of units from a large corpus requires determining the optimal weighting of various acoustic factors (Black and Campbell, 1995). In one approach to the problem, an utterance is synthesized from the best set of units in the database, and its distance from the natural waveform is measured—the process iterates over different weight settings until the best set of weight values is determined (Black and Campbell, 1995).
2. Depending on which "non-native" sounds found in loanwords are included, writers differ on the exact number of vowels and consonants that comprise the phoneme inventory of Hindi.
3. We represent multicharacter symbols with braces around them; thus [aa] in "k[aa]n" ('ear') corresponds to the phoneme represented by the IPA symbol /a/.
4. Following linguistic convention, the asterisk in *[mahaa[++] ngar] indicates that the form is unattested
5. The actual rule in our TTS system was slightly more complicated, since we defined our rules over two intermediate levels of orthographic representations.
6. Long vowels, retroflex and aspirated stops and flaps are represented with multicharacter symbols enclosed in square brackets, as in the long vowel '[ii]' and the voiced aspirated afficate '[jh]'.
7. Three-member compounds are rare and hence not included.
8. We did not count as correct those instances where schwa-deletion did not occur in inappropriate contexts.

References

Bhaskararao, P. and Mathew, S. (1992). Phonemic transcription rules for text-to-speech synthesis of Hindi. In R.M.K. Sinha (Ed.), *Computer Processing of Asian Languages*. New Delhi: Tata McGraw Hill.

Bhaskararao, P., Peri V.N., and Udpikar, V. (1994). A text-to-speech system fo application by visually handicapped and illiterate. *Proceedings of the International Conference on Spoken Language Processing*, pp. 1239–1241.

Black, A. and Campbell, N. (1995). Optimising selection of units from speech databases for concatenative synthesis. *Proceedings of Eurospeech 95*. Madrid, Spain, Vol. 1, pp. 581–584.

Charpentier, F. and Moulines, E. (1990). Pitch-synchronous waverform processing techniques for text-to-speech synthesis using diphones. *Speech Communication, 9*(5/6):453–467.

Furtado, X.A. and Sen, A. (1996). Synthesis of unlimited speech in Indian languages using formant-based rules. Sãdhanã, *21*:345–362.

Hopcroft, J. and Ullman, J. (1979). *Introduction to Automata Theory, Languages, and Computation*. Reading, MA: Addison Wesley.

Jannedy, S. and Möbius, B. (1997). Name pronunciation in German text-to-speech synthesis. *Proceedings of the 5th Conference on Applied Natural Language Processing*. Washington, DC, pp. 49–56.

Kachru, Y. (1987). Hindi-Urdu. In B. Comrie (Ed.), *The World's Major Languages*. London & Sydney: Kroom Helm.

Kaplan, Ronald and Kay, Martin. (1994). Regular models of phonological rule systems. *Computational Linguistics, 20*:331–378.

Kiraz, G. and Möbius, B. (1998). Multilingual syllabification using weighted finite-state transducers. *Proceedings of the Third International Workshop on Speech Synthesis*. Jenolan Caves, Australia, pp.71–76.

Klatt, D.H. (1980). Software for a cascade/parallel formant synthesizer. *Journal of the Acoustical Society of America, 67*(3):971–994.

McGregor, R. (1995). *Outline of Hindi Grammar*. Oxford: Oxford University Press.

Möbius, B. (1999). The Bell Labs German text-to-speech system. *Computer Speech and Language*, 13:319–358.

Möbius, B. (2000). Corpus-based speech synthesis: Methods and challenges. *Arbeitspapiere des Instituts für Maschinelle Sprachverarbeitung (Univ. Stuttgart)*, AIMS, 6(4):87–116.

Möbius, B. and van Santen, J. (1996). Modeling segmental duration in German text-to-speech synthesis. *Proceedings of the Fourth International Conference on Spoken Language Processing*. Philadelphia, PA, Vol. 4, pp. 2395–2399.

Möbius, B., Schroeter, J., van Santen, J., Sproat, R., and Olive, J. (1996). Recent advances in multilingual text-to-speech synthesis. *Fortschritte der Akustik - DAGA 96*. DEGA, Oldenburg, pp. 82–85.

Möbius, B., Sproat, R., van Santen, J., and Olive, J. (1997). The Bell Labs German text-to-speech system: An overview. *Proceedings of the European Conference on Speech Communication and Technology*. Rhodes, Greece, Vol. 5, pp. 2443–2446.

Mohri, M. (1997). Finite-state transducers in language and speech processing. *Computational Linguistics*. 23:2.

Mohri, M., Pereira, F., and Riley, M. (1998). A rational design for a weighted finite-state transducer library. In D. Wood and S. Yu (Eds.), *Automata Implementation, Lecture Notes in Computer Science 1436*. Berlin-NY: Springer Verlag, pp. 144–158.

Mohri, M. and Sproat, R. (1996). An efficient compiler for weighted rewrite rules. *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*. Santa Cruz, CA, pp. 231–238.

Ohala, M. (1983). *Aspects of Hindi Phonology*. Delhi: Motilal Banarsidass.

Ohala, M. (1999). Hindi. *Handbook of the International Phonetic Association: A guide to the Use of the International Phonetic Alphabet*. Cambridge: Cambridge University Press, pp.100–103.

Pandey, P. (1989). Word accentuation in Hindi. *Lingua, 77*(1):37–73.

Rao, P.V.S. (1993). VOICE: An integrated speech recognition synthesis system for the Hindi language. *Speech Communication*, 13:197–205.

Sen, A. and Samudravijaya, K. (2002). Indian accent text-to-speech system for web browsing. *Sādhanā*, 27(1):113–126.

Shih, C. and Sproat, R. (1996). Issues in Text-to-Speech Conversion for Mandarin. Computational Linguistics and Chinese Language Processing.

Singh, R. and Agnihotri, R. (1997). *Hindi Morphology: A Word-Based Description*. Delhi: Motilal Banarsidass.

Sproat, R. (1996). Text interpretation for TtS synthesis. In R. Cole (Ed.), *Survey of the State of the Art in Human Language Technology* (http://cslu.cse.ogi.edu/HLTsurvey/).

Sproat, R. (1997). Multilingual text analysis for text-to-speech synthesis. *Natural Language Engineering*. 2(4):369–380.

Sproat, R. (Ed.). (1998). *Multilingual Text-to-Speech Synthesis*. Dordrecht: Kluwer Academic Publishers.

Van Santen, J. (1994). Assignment of segmental duration in text-to-speech synthesis. *Computer Speech and Language, 8*:95–128.

Van Santen, J. (1998). Timing. In R. Sproat (Ed.), *Multilingual Text-to-Speech Synthesis*. Dordrecht: Kluwer Academic Publishers, pp.115–139.

Van Santen, J., Shih, C., and Möbius, B. (1998). Intonation. In R.Sproat (Ed.), *Multilingual Text-to-Speech Synthesis*. Dordrecht: Kluwer Academic Publishers, pp.115–139.

Verma, R., Sarma, S.S.A., Shrotriya, N., Sharma, A.K., and Agrawal, S.S. (1995). On the development of text to speech system for Hindi. *Proceedings of the International Congress of Phonetic Sciences*. Stockholm, Sweden, pp. 354–357.