A Rule Based Schwa Deletion Algorithm for Hindi

Monojit Choudhury
Anupam Basu
Dept of Computer Science & Engineering.
Indian Institute of Technology,
Kharagpur - 721302
West Bengal, India.
Email: <*monojit, anupam*>*@cse.iitkgp.ernet.in*

Abstract

This paper describes the phenomenon of schwa deletion in Hindi and proposes a rule-based algorithm for solving the problem, which is required for a concatenative Text-to-Speech (TTS) system for Hindi. We show here that the algorithm gives correct results for 96.12% of the common Hindi words. We also show that the performance can be improved further with the help of a morphological analyzer. We have compared our work with the previous one in this field, highlighting the differences and the advantages of our approach over theirs. Finally, we have tried to explain the rules and significance of the algorithm from a linguistic perspective using the concepts of syllable economy, phonotactic constraints and word morphology.

1. Introduction

A *Text to Speech* (*TTS*)[Dutoit, 1996; Allen *et al.*, 1987] system converts a written text to speech or a sound file. The goal of a *TTS* system is to provide intelligible speech, which is as natural as possible. Mainly, a *TTS* system consists of a *Natural Language Processing (NLP)* module and a *Digital Signal Processing (DSP)* module. The *NLP* module converts the text that is the *graphemes* to a string of *phonemes*. It also encodes the *intonation* and *prosodic* information in the output string. The *DSP* module obtains the sound files from an acoustic inventory corresponding to the string of *phonemes* or *diphones*[1] and concatenates them. Finally, it modulates the sound according to the *intonation* and *prosodic* information. Though *intonation* and *prosody* add mainly to the naturalness of the speech, certain linguistic analysis becomes necessary even for the intelligibility of the speech, an essential quality of any *TTS* system. This is because in almost all languages, we hardly pronounce what we write.

[1] The acoustic inventory may store phonemes, diphones, triphones, or even syllables. This partly depends on the language and partly on the technique chosen for synthesis.

Hindi is written from left to right using the *Devanagari* script. Having its root in Sanskrit, which is phonetically perfect (i.e. there is very little or almost no discrepancy between written text and pronunciation), Hindi is pronounced almost as it is written. Going by the *Optimality Theory* [Kager, 1999], this can be restated as "In Sanskrit and Hindi the *faithfulness constraints* are ranked higher than the *markedness constraints*". Each consonant in written Hindi is associated with an "inherent" *schwa*[2], which is not explicitly represented. Other vowels are overtly written diacritically or non-diacritically around the consonant. The problem is that *schwa* is sometimes pronounced and sometimes not. For example, in the word ***dha.DakaneM***[3] [धड़कनें, dhə ɽ .kə n.ẽ , noun. heart-beats], the *schwa* following ***.D*** is deleted in the pronunciation. Just to illustrate how improper *schwa deletion* can really render the speech incomprehensible, compare the above word with ***dha.Dakane*** [धड़कने, dhə .ɽ ə k.ne, verb. To beat (heart), with *case-ending* ***ne***], where *schwa* following ***k*** is deleted. Without any schwa deletion, not only the two words will sound very unnatural, but it will also be extremely difficult for the listener to distinguish between the two, the only difference being nasalization of the ***e*** at the end of the former. However, a native speaker would pronounce the former as *dha.D-kan-eM* and the later as *dha-.Dak-ne*, which are clearly distinguishable. Thus, any *TTS* system for Hindi must have an efficient schwa deletion module.

In this paper, we describe a rule-based *schwa deletion* algorithm for Hindi. Section 2 lists out some of the empirical observations about contexts where *schwas* are either invariably deleted or retained. However, these rules, as we shall see, do not span over all possible contexts making it necessary to design some additional mechanism for predicting the nature of all the *schwas* present in a word. Section 3 describes such an algorithm that will finally decide which *schwas* are to be deleted. Section 4 shows how morphology of a word affects the syllable structure and hence the *schwa deletion*. Section 5 is devoted to performance analysis of the algorithm and section 6 compares our algorithm with related previous works. In this section, we have also suggested two slightly different variants of the algorithm. Section 7 concludes this paper by discussing possible linguistic explanations behind the algorithm and throws some light over current research and scope for future work in this direction.

2. Empirical Observations

Simple observation of Hindi words provides us with certain contexts where *schwa* is retained and certain contexts where it is deleted without any exception. Some of these contexts of *schwa* retention arise from *phonotactic constraints* and certain contexts for deletion, though are not phonetically obvious, are confirmed by empirical observations. These contexts have been listed below along with illustrations and *phonetic* reasoning (wherever applicable).

1. The schwa of a syllable immediately followed by a conjugate syllable (*yuktakshara*) is always retained. For example in ***sAphalya*** [साफल्य, sa.ɸ ə l.jə , success] and ***AmantraNa*** [आमंत्रण, a.mə n.trə ɳ , invitation] the *schwas* following ***ph*** and ***m*** are retained as they are followed by the conjugate syllables ***lya*** and ***ntra*** respectively. This rule is

[2] The first vowel of Hindi alphabet, अ (pronounced as ə or ʌ , but for our convenience we shall denote it as ə only for both the contexts).

[3] In this paper, Hindi graphemes are written using roman script following the ITrans convention. However, the graphemes of the word in *Devnagari* script, its pronunciation in *International Phonetic Alphabet* (*IPA*) and the meaning in English are given within the parentheses immediately following the word. Occasionally we have represented the pronunciation using roman script to clearly illustrate the schwas which are deleted and those which have been retained. In such cases, '-' marks the syllable breaks.

partially due to *phonotactic constraints*. For example in the above two cases, if the *schwa* following ***ph*** or ***m*** is deleted, then we end up with consonant clusters ***phly*** or ***mntr***, which are impossible to pronounce. Deletion of *schwa* in such a context might not always lead to an illegal consonant cluster; even then, this rule can be generalized. Note that for non-monomorphemic words this rule may not be applicable. Section 4 discusses such cases.

2. If *y* (य) is followed by the inherent *schwa* and preceded by a syllable with a high vowel such as ***i***, ***I***, ***R^i, u*** or ***U*** then the schwa following ***y*** is always retained. For example in ***priya*** [प्रिय, pri.jə , beloved] and ***tR^itIya*** [तृतीय, tri.tiː .jə , third]. On the other hand for low and medium height vowels like ***a***, ***A***, ***e*** or ***o***, the *schwa* following ***y*** may be deleted. For example in ***Aya*** [आय, aĕ , income] and ***hoya*** [होय, hoĕ , happens]. The consonant ***y*** is a glide from high vowel to a medium vowel. Therefore, if the schwa is deleted in the context where ***y*** follows a high vowel, the glide will be lost resulting in absence of ***y*** in the pronunciation. However, if ***y*** is preceded by a low or medium vowel, the deletion of schwa still maintains a glide from the previous vowel to a higher vowel which makes the presence of ***y*** discernable.

3. Any conjugate syllable or cluster of consonants that ends in (i.e. the last consonant of the cluster/syllable is) ***y***, ***r***, ***l*** or ***v***, the *schwa* following the cluster is retained. For example in ***kAvya*** [काव्य, kaw.jə , poetry], ***samprati*** [सम्प्रति, səm.prə.ti, recently], ***ashva*** [अश्व, ə ʃ .wə , horse] and shukla [शुक्ल, ʃ uk.lə , white] the *schwas* following ***y***, ***r***, ***l*** and ***v*** are retained. This is also due to *phonotactic constraint*.

4. The schwa preceding a full vowel[4] is retained to maintain lexical distinctions. For example in the word ***ba.DhaI*** (बढ़ई, bə .ɽ hə iː , carpenter] the schwa following ***.Dh*** is retained since otherwise the resulting word would be indistinguishable from the word ***ba.DhI.***

5. The schwa following ***h*** is always retained like in the words ***samuha*** [समुह, sə .mu.ha, group] and ***cheharA*** [चेहरा, ce.he.ra, look].

6. The *schwa* of the first syllable is never deleted. For example, the *schwas* following ***b*** in ***badarA*** [बदरा, bə d.ra, cloud], ***k*** in ***kalama*** [कलम, kə .lə m, pen] or ***Sh*** in ***kShamatA*** [क्षमता, kʂ ə m.ta, ability] are retained. Deletion of the *schwa* in the first syllable can not only result in illegal consonant clusters, but can also change the identity of the word.

7. If the last syllable of the word contains a *schwa* and contexts 1 through 6 described above for the retention of the *schwa* do not occur then the *schwa* is to be deleted. For example, the *schwas* following ***m*** in ***kalama, d*** in ***banda*** [बंद, bə nd, closed] or ***k*** in ***tarka*** [तर्क, tə rk, argument] are deleted.

Whether a *schwa* will be retained or deleted can be clearly determined in the contexts described above, but we cannot conclude about the deletion of other schwas, which do not pertain to any of these contexts. For example, in the word ***bachapa***na [बचपन, bə c.pə n, childhood] we can infer that *schwa* following ***b*** will be retained (context 6), whereas that following ***n*** will be deleted (context 7). However, we are not in a position to conclude anything re-

---

[4] Vowels can occur in two forms – full or *maatraas*. E.g. in the word ***AnA*** [आना, a.na, to come], the first ***A*** is a full vowel whereas the second one is a *mAtrA*

garding the *schwas* following ***ch*** and ***p***. Next two sections, describe an algorithm for determining the behavior of such *schwas*.

3. The Algorithm

For description of the algorithm, we shall take the help of a notation called *half* ($\mathcal{H}$) and *full* ($\mathcal{F}$) sounds. We define a *full* sound as a consonant-vowel pair or a vowel alone, whereas *half* sound as a pure consonant sound, without any vowel modulation (*mAtrA*). Therefore, any vowel or a consonant followed by a vowel (*mAtrA*) is a *full* sound, whereas a consonant followed by *halant* (i.e. the consonants of a conjugate syllable or cluster, except the last one) are *half* sounds. Hence, when a *schwa* following a consonant is deleted, it becomes *half*, but if it is retained, the consonant is *full*. Since the nature of the consonants followed by *schwa* might not be known beforehand, we shall call such consonants as *unknown* ($\mathcal{U}$). To illustrate this point, consider the example of ***bachapana*** cited before. Here, ***b*** is $\mathcal{F}$, ***n*** is $\mathcal{H}$ but ***ch*** and ***p*** are $\mathcal{U}$. In the algorithm, only the consonants and full vowels will be marked $\mathcal{H}$, $\mathcal{F}$ or $\mathcal{U}$, but the *mAtrAs* will not be marked. After marking the consonants of the word according to the rules stated in the last section, only the consonants followed by *schwas* can be marked as $\mathcal{U}$. The algorithm then scans the marked word from left to right replacing each of the $\mathcal{U}$s by either $\mathcal{F}$ or $\mathcal{H}$, depending on the two adjacent syllables of that particular $\mathcal{U}$-marked consonant. The basic idea here is to minimize the number of syllables in the word by deleting as many schwas as possible without violating any *phonotactic constraints*, which requires retention of those *schwas* which have an $\mathcal{H}$-marked consonant as at least one of its neighbors. At the end of the algorithm, *schwas* following the consonants marked as $\mathcal{H}$ are deleted.

The formal steps of the algorithm are described next. Figure 1 illustrates the stepwise execution of the algorithm on the words ***bachapana***, ***priyatama*** [प्रियतम, pri.jə .tə m, beloved] and ***AmantraNa***.

procedure *delete_schwa(DS)*

Input: word: string of alphabets (graphemes)[5]
Output: input word with some of the schwas deleted.

1. Mark all the full vowels and consonants followed by vowels other than the inherent *schwas* (i.e. consonants with *mAtrAs*) and all the ***h***s in the word as $\mathcal{F}$ unless it is explicitly marked as *half* by use of *halant*. Mark all the consonants immediately followed by consonants or *halants* (i.e. consonants of conjugate syllables) as $\mathcal{H}$. Mark all the remaining consonants, which are followed by implicit schwas as $\mathcal{U}$.
2. If in the word, ***y*** is marked $\mathcal{U}$ and preceded by ***i***, ***I***, ***ri***, ***u*** or $\mathcal{U}$ mark it $\mathcal{F}$ (context 2).
3. If ***y***, ***r***, ***l*** or ***v*** are marked $\mathcal{U}$ and preceded by consonants marked $\mathcal{H}$, then mark them $\mathcal{F}$ (context 3).
4. If a consonant marked $\mathcal{U}$ is followed by a full vowel, then mark that consonant as $\mathcal{F}$ (context 4).
5. While traversing the word from left to right, if a consonant marked $\mathcal{U}$ is encountered before any consonant or vowel marked $\mathcal{F}$, then mark that consonant as $\mathcal{F}$ (context 6).

[5] In order to keep the description of the algorithm simpler, the output is also presented as a string of graphemes instead of phonemes. After *schwa deletion*, grapheme to phoneme mapping for Hindi becomes quite simple.

6. If the last consonant is marked 𝒰, mark it ℋ (context 7).
7. If any consonant marked 𝒰 is immediately followed by a consonant marked ℋ, mark it ℱ (context 1).
8. While traversing the word from left to right, for every consonant marked 𝒰, mark it ℋ if it is preceded by ℱ and followed by ℱ or 𝒰 otherwise mark it ℱ.
9. For all consonants marked ℋ, if it is followed by a schwa in the original word, then delete the schwa from the word. The resulting new word is the required output.

End procedure *delete_schwa*

Word	*ba-cha-pa-na*	*p-ri-ya-ta-ma*	*A-ma-n-t-ra-Na*
After Step			
1	𝒰--𝒰--𝒰--𝒰	ℋ-ℱ-𝒰-𝒰-𝒰	ℱ-𝒰-ℋ-ℋ-𝒰-𝒰
2	𝒰--𝒰--𝒰--𝒰	ℋ-ℱ-ℱ-𝒰-𝒰	ℱ-𝒰-ℋ-ℋ-𝒰-𝒰
3	𝒰--𝒰--𝒰--𝒰	ℋ-ℱ-ℱ-𝒰-𝒰	ℱ-𝒰-ℋ-ℋ-ℱ-𝒰
4	ℱ--𝒰--𝒰--𝒰	ℋ-ℱ-ℱ-𝒰-𝒰	ℱ-𝒰-ℋ-ℋ-ℱ-𝒰
5	ℱ--𝒰--𝒰--𝒰	ℋ-ℱ-ℱ-𝒰-𝒰	ℱ-𝒰-ℋ-ℋ-ℱ-𝒰
6	ℱ--𝒰--𝒰--ℋ	ℋ-ℱ-ℱ-𝒰-ℋ	ℱ-𝒰-ℋ-ℋ-ℱ-ℋ
7	ℱ--𝒰--𝒰--ℋ	ℋ-ℱ-ℱ-𝒰-ℋ	ℱ-ℱ-ℋ-ℋ-ℱ-ℋ
8.1	ℱ--ℋ--𝒰--ℋ	ℋ-ℱ-ℱ-ℱ-ℋ	--
8.2	ℱ--ℋ--ℱ--ℋ	--	--
Results: 9	*bach-pan*	*pri-ya-tam*	*A-man-traN*

Figure 1 Illustration of the working of the algorithm

4. Need for Morphological Analysis

Algorithm *delete_schwa* (*DS*), as described in section 3 gives erroneous results in many cases. Consider the word ***dha.Dakane*** as cited in the introductory section. After the first 6 steps of the algorithm, the consonants will be marked as ℱ𝒰𝒰ℋ, but in the 7th step the two unknown schwas following ***.D*** and ***k*** will be marked as ℋ and ℱ respectively. After schwa deletion the pronunciation will be *dha.D-ka-ne*. However, the actual pronunciation is *dha-.Dak-ne*. Similar is the case with the word ***asamaya*** [असमय, ə .sə .mə ĕ , untimely], which is modified to *as-may* by the algorithm whereas the real pronunciation is *a-sa-may*.

The reason for these discrepancies is that the words cited above are not *monomorphemic*. ***dha.Dakane*** is derived from the root verb ***dha.Daka*** [धड़क, dhə .ɽ ə k] by juxtaposing the case-ending (called *vibhakti* in Hindi) ***ne*** [ने]. When the algorithm is applied to ***dha.Daka***, it gives *dha-.Dak* and juxtaposing the ***ne*** gives *{dha-.Dak-ne*, which is the correct pronunciation. Similarly ***asamaya*** is composed of ***a*** (अ, ə , a negative prefix) and ***samaya*** [समय, sə .mə ĕ , time), which when separately modified by the algorithm *DS* and juxtaposed gives *a-sa-may*, the correct pronunciation.

Thus, the root sound remains unchanged even if it is juxtaposed or modified by case-endings, suffixes or prefixes or other words (in case of compound or conjugate words i.e. *samaasa* or *sandhi*). This tendency results from *faithfulness* to the lexical conventions and has important role to play in intelligibility of the speech. Thus, a module for *morphological analysis* of the words is required before applying the algorithm. A possible design of a morphological analyzer for Hindi, which is now in a developing stage, has been briefly outlined in the concluding section. Here, we describe the various word-formation rules in Hindi, and their affect on the syllable structure.

1. *Samaasa* or compound words are formed simply by concatenation of two smaller words. Each of the words retains their original pronunciation, so the *schwa* deletion module is applied separately on the words and the results are simply juxtaposed to get the pronunciation of the compound word. For example, ***charaNakamala*** [चरणकमल, cə .rə ɳ .kə .mə l] => (after *morphological analysis*) ***charaNa*** [foot] & ***kamala*** [lotus] => (after individual *schwa deletion*) *cha-raN* & *ka-mal* => (after juxtaposition, final result) *cha-raN-ka-mal*. (On the other hand, without *morphological analysis*, the result would have been *char-Nak-mal*, which is wrong.)

2. For *Upasarga* or prefixes, which are juxtaposed before the root word, the rule is identical to that for *samaasa*. E.g. ***pragati*** [प्रगति, prə .gə .ti, development] or ***asamaya***.

3. *Pratyaya* or suffixes which begin with a consonant are simply juxtaposed at the end of the stem as in rule 1 above. However, if the suffix begins with a vowel, schwa deletion algorithm (*DS*) is applicable to the whole word instead of the stem and the suffix separately. For example, ***arabI*** [अरबी, ə r.biː , Arabian] is made up of the root ***arab*** and suffix ***I***, but the pronunciation is *ar-bi* (and not *a-ra-bi*). Similarly, ***namakIna*** [नमकीन, nə m.kiː n, salty] is derived from the ***namaka***(salt) and suffix ***In***, but the pronunciation is *nam-kIn* (and not *na-ma-kin*).

4. Stems, which have a conjugate syllable in the second last position, are exceptions to rule 3. For words derived from such stems, schwa deletion is separately applicable to the stem and the affixes. For example, ***nindokoM*** [निंदकों, nin.də k.õ , critics, with plural marker ***oM***] is pronounced as *nin-da-koM* and not *nind-koM*.

5. In *sandhi* or conjugate words the pronunciation of original words are maintained except at the junction of the *sandhi*, where the pronunciation depends on the type of the *sandhi,* governed by the orthographic rules. A detailed analysis of the rules for *sandhi* is beyond the scope of this paper. Nonetheless, decomposing the word into its sub-parts and deleting the internal *schwas* according to the algorithm *DS*, and at the junction retaining the *schwas*, if any, solves the purpose.

The above rules can be used to modify the algorithm *DS* as follows.

Procedure *modified_delete_schwa*(*MDS*)

Input: word: string of alphabets
Output: input word with some of the *schwas* deleted.

1 Analyze the *morpheme* boundaries of the word using *morphological analyzer*.

2 If the word is not *monomorphemic*, then classify the word depending on its morphology to one of the above classes (1 to 5)
3 If the word is *monomorphemic*, apply algorithm *DS*,
else apply algorithm *DS* to the individual morphemes, as suggested by the rules above and concatenate the outputs accordingly.

End Procedure *modified_delete_schwa*

5. Performance Analysis

The algorithm *DS* has been implemented in C and integrated with iLEAP, a software supporting Indian language fonts. All the Hindi words in a pocket dictionary ["Hindi Bangla English – Tribhasa Abhidhaan", Sandhya Publication, 1st Edition March 2001] were tested for *schwa deletion*. The output was checked manually. Since *morphological analyzer* has yet not been fully realized, compound and *non-monomorphemic* words were tested manually in a separate experiment, by dry run of the algorithm *MDS* over all the words, where it was assumed that the morphological analyzer always gives the correct decomposition of the word. The results of the experiments are as follows.

Without *Morphological Analysis* (i.e. the algorithm *DS*)
Total number of words tested: 11095
Number of words with wrong *schwa deletion* results: 431
Thus, correctness of the algorithm: 96.12%

With a *Morphological Analyzer* (i.e. the algorithm *MDS*)
Total number of words tested: 11095
Number of words with wrong *schwa deletion* results: 12
Thus, correctness of the algorithm: 99.89%

Morphology of the word	Accounting for % error
Pratyaya or Suffix	10.20
Upasarga or Prefix	13.69
SamAsa or Compoundation	35.73
Sandhi or Conjugation	7.42
Vibhakti or Case endings	30.16
Others	2.78

Table 1: The breakup of the morphology of the word accounting for the error in delete_schwa algorithm
[The rules for handling case endings and suffixes are same, though they have been shown as separate classes]

Some of the words for which *MDS* gave wrong results are ***khataranAka*** [खतरनाक, khə .tə r.nak, dangerous], ***kadalI*** [कदली, kə .də .liː , banana], ***Anayana*** [आनयन, a.nə .jə n, the act of bringing] etc. Table 1 gives the breakup of type of the morphology of the words resulting in incorrect *schwa deletion* when a *morphological analyzer* is not used (i.e. for *DS*). It should be noted that since a dictionary does not list all the inflected forms of a word and moreover frequency of occurrence of the words in normal texts have been neglected in the analysis, the performance of *DS* seems to be overestimated. However, MDS is expected to be highly accurate even when tested with normal text or corpora. Such experiments are going on and will be reported subsequently.

6. Previous Works and Other Variants of The Algorithm

The problem of *schwa deletion* exists in many languages like French, Dutch, English or Bengali. Unlike Hindi, in some languages like Dutch and French, *schwa deletion* is optional and depends on the context and the speaker. Substantial amount of work has been done on the computational aspects of schwa deletion in these languages [Travel and Bernard, 1999; Fourgereon, 1997]. Although the problem of *schwa deletion* has been addressed from a linguistic perspective [Kaira, 1976], for Indian languages, very little work has been done on the computational aspects. The only computational model for *schwa deletion* in Hindi that we could locate was by B. Narsimhan et al [Narsimhan *et al.*, 2001].

Their work combines Ohala's work (1983) and *morphological analysis* with *finite state transducers* [Kaplan and Kay, 1994] and cost models. Although at a fundamental level the concepts are not poles apart, but the approach is altogether different from ours. Their work to some extent is based on the *Optimality Theory* [Kager, 1999]. Initially the algorithm generates all possible output candidates for a given input, following Ohala's work on possible contexts for *schwa deletion*. Then certain candidates, which violate *phonotactic constraints*, are filtered out. Among the remaining candidates, the one with the minimum cost according to the cost model is selected as the final output. The advantages of a rule-based method like ours over their approach lies in its simplicity and ease of computation. Any rule-based algorithm scans the input-word locally for contexts where a rule is applicable and if any of the contexts arises, the rule is applied and the word is passed on to the next rule. Thus, the number of times the word is scanned is at most equal to the number of rules. If the rules are independent, the algorithm can be efficiently implemented to reduce the number of scans. In fact, *DS* needs to scan a word only twice. On the other hand, generation of all the possible candidate words and searching the whole solution space is definitely more computationally expensive. Secondly, many a times, rules are capable of capturing the underlying theory for such a phenomenon in a more straightforward manner.

It may be very tough in general, to discover the rules involved in certain linguistic phenomenon like *schwa deletion*, neither might it provide a general computational framework for all the languages, even then once a rule-based algorithm succeeds in solving a problem, it is definitely going to outperform any search based method on computational grounds. The pros and cons of our algorithm as compared to [Narsimhan *et al.*, 2001] are summarized below.

- Advantages:
  1. Simple to implement and straightforward.
  2. Less computation required, hence better *throughput*.
  3. Better overall performance (gives correct results for 99.89% words compared to 89% for their algorithm).
- Disadvantages:
  1. Rules cannot be generalized for other languages.
  2. Listing out the rules requires extensive observation of the words, which is a tedious job.

Another notable difference between our work and [Narsimhan *et al.*, 2001] is that we apply the *schwa deletion* algorithm from left to right (step 8 of the algorithm *DS*) whereas they have chosen to apply the rules from right to left for intra-morphemic *schwa deletion*. This gives rise to an interesting question of what happens if we apply our rule from right to left, instead. In fact, doing so we land up in a slightly different variant of the algorithm, which we call the *reverse_delete_schwa* (*RDS*). Statistical analysis has shown that *DS* performs substantially better than *RDS*, though *modified_reverse_delete_schwa* (*MRDS*) and *MDS* does not differ much in performance. However, a hybrid variant of the two called *HDS* can be designed to account for many more cases, but the extra effort is not of much worth when compared to the gain in performance. Since *MDS* alone can give 99.89% correct results, it is a better option to prepare a small exception list for *MDS* to handle all the cases rather than going for *HDS*, which is much more computationally intensive.

It is to be noted that one similarity between our approach and [Narsimhan *et al.* 2001] is the use of a *morphological analyzer*.

7. Conclusion and Future Works

We have described a rule-based algorithm for *schwa deletion* and some of its variants. We have also seen that the performance of the algorithm is upgraded by using *morphological analysis* of words. Currently, we are developing a *morphological analyzer* for Hindi for which instead of a linear lexicon, we propose to use a *WordNet* like structure consisting information about the root, possible prefixes, suffixes and case endings for a word, along with conventional relations like synonymy and antonymy. Interestingly, question may arise that if we store all the words in a wordnet, then why not use a little more memory to store information regarding pronunciations of the words so that we do not need any *schwa deletion* algorithm at all. However, there are many reasons in favor of having a *schwa deletion* algorithm, such as:

1. We need a *WordNet* for root words only, which is at least 10 times smaller than an all word lexicon for Hindi.
2. Rich word forming techniques like *samAsa* and *sandhi* in Hindi provides the speaker the freedom of forming new words. Therefore, it becomes impossible to store all possible words.
3. New foreign words fuse into the lexicon every now and then, which adds to the above reason. It goes true for large number of proper nouns too.
4. As the size of the wordnet becomes larger, both memory requirements and searching time increase. Therefore, if we wish to develop a portable *TTS* system, these issues create real hurdles, which can be overcome by using algorithms like *DS* or *MDS*.

Finally, before concluding this paper, let us have a closer look on the algorithm from a phonological perspective. It is a well known fact, the so called *pleasure principle*; that in all languages there is a tendency to reduce the work done by lips and tongue while speaking. There are several means of reducing the amount of work, one of them being schwa deletion, which reduces the number of syllables in the word. The tendency of schwa deletion is very high in Hindi. A schwa is deleted as long as it does not violate any *phonotactic constraints* (i.e. does not create any unutterable cluster of consonants, i.e. *markedness constraints*) and as long as the speech remains intelligible (*faithfulness constraints*). The former has lead to the rules described in section 2, whereas the later has made the morphological analysis of the words necessary. After applying these two basic rules, we shall end up in a set of schwas, which can be deleted. However, here we have to make a choice of which *schwas* are to be deleted without creating illegal consonant clusters. *DS* and *RDS* suggest two different methods of choosing those *schwas*. They are two methods of minimizing the number of syllables in a word subject to the constraints stated above. When *DS* and *RDS* give different results, it means that there are two possibilities only one of which is acceptable. It should be added here that for some words like ***kadalI*** both of these algorithms fail, as none of the *schwas* are deleted. It will be an interesting research to study the linguistic reasons underlying such phenomena.

The algorithm described here can have applications in *schwa deletion* for other Indian languages like Bengali on which work has already started. *Intonation* and *prosodic* modeling are the other two fields in which research is going on for the development of natural and intelligible *TTS* for Hindi and Bengali.

Acknowledgement

This research was supported by *Media Labs Asia* research funding. We are grateful to Mrs. Samhita Deb and Prof. Jayshree Chakrabarty for their useful comments and suggestions on the linguistic aspects of this paper. We are also thankful to Mr. Mohit Kumar who helped in implementing the algorithm.

References

[Allen *et al.*, 1987] Allen J, Hunnicut S and Klatt D. From Text To Speech, The MITTALK System. *Cambridge University Press*, 1987.

[Dutoit, 1996] Dutoit T. An Introduction to Text-To-Speech Synthesis. *Kluwer Academic Publishers*, 1996.

[Fourgereon, 1997] Fourgereon, Cecile, and Steriade D. Does deletion of French schwa lead to neutralization of lexical distinctions? In *Proceedings of Euro-speech 97*, Vol. 2, pp.943-946

[Kager, 1999] Kager R. Optimality Theory. *Cambridge University Press*, 1999

[Kaira, 1976] Kaira S. Schwa-deletion in Hindi. In *Language forum (back volumes), Bhari publications*, Vol. 2, No. 1, April-June 1976

[Kaplan and Kay, 1994] Kaplan R. M and Kay M. Regular models of phonological rule systems. In *Computational Linguistics*, Vol. 20, no. 3, pp. 331-378, Sept. 1994

[Narsimhan *et al.*, 2001] Narasimhan B, Sproat R, and Kiraz G. Schwa-deletion in Hindi Text-to-Speech Synthesis. In *Workshop on Computational Linguistics in South Asian Languages*, 21st SALA, October 2001, Konstanz

[Travel and Bernard, 1999] Travel and Bernard. Optional Schwa Deletion: on syllable economy in French. In *Formal Perspectives on Romance Linguistics*, Ed. By J. Mark Authier, Barbar S. Bullock, & Lisa A. Reed., 1999