TOOLS FOR THE DEVELOPMENT OF A HINDI SPEECH SYNTHESIS SYSTEM

Kalika Bali, Partha Pratim Talukdar
N. Sridhar Krishna

HP Labs India
24 Salarpuria Arena, Hosur Road
Bangalore, India
{kalika, partha.talukdar,
nemala-sridhar.krishna}@hp.com

A.G. Ramakrishnan

Department of Electrical Engineering
Indian Institute of Science, Bangalore
India
{ramkiag}@ee.iisc.ernet.in

ABSTRACT

We describe in detail a Grapheme-to-Phoneme (G2P) converter required for the development of a good quality Hindi Text-to-Speech (TTS) system. The Festival framework is chosen for developing the Hindi TTS system. Since Festival does not provide complete language processing support specific to various languages, it needs to be augmented to facilitate the development of TTS systems in certain new languages. Because of this, a generic G2P converter has been developed. In the customized Hindi G2P converter, we have handled schwa deletion and compound word extraction. In the experiments carried out to test the Hindi G2P on a text segment of 3485 words, 97.36% word phonetisation accuracy is obtained. This Hindi G2P has been used for phonetising large text corpora which in turn is used in designing an inventory of phonetically rich sentences. The sentences ensured a good coverage of the phonetically valid diphones using only 0.3% of the complete text corpora.

1. INTRODUCTION

Hindi, the official language of India, is spoken as a first language by 33 percent of the Indian population, and by many more as a lingua franca. In contrast, only a very small percentage of Indians use English as a means of communication. This fact, coupled with the prevalent low literacy rates make the use of conventional user interfaces difficult in India. Spoken language interfaces enabled with Text-to-Speech synthesis have the potential to make information and other ICT based services accessible to a large proportion of the population.

However, good quality Hindi TTS systems that can be used for real time deployment are not available. Though a number of research prototypes of Indian language TTS systems have been developed [1] [2], none of these are of quality that can be compared to commercial grade TTS systems in languages like English, German and French. The foremost reason for this is that developing a TTS system in a new language needs inputs for resolving language specific issues requiring close collaboration between linguists and technologists. Large amount of annotated data is required for developing language-processing modules like parts of speech (POS) taggers, syntactic parsers, intonation and duration models. This is a resource intensive task, which is prohibitive to academic researchers in emerging economies like India.

In this paper, we describe the effort at HP Labs India to develop a Hindi TTS system based on the open source TTS framework, Festival [3]. This effort is a part of the Local Language Speech Technology Initiative (LLSTI) [4], which facilitates collaboration between motivated groups around the world, by enabling sharing of tools, expertise, support and training for TTS development in local languages. LLSTI aims to develop a TTS framework around Festival that will allow for rapid development of TTS systems in any language.

The focus of this paper is on a Hindi G2P converter and on the design of phonetically balanced sentences. Certain linguistic characteristics of the Hindi language, like a large phone set and the schwa deletion issue, pose problems for developing unit selection inventories and Grapheme-to-Phoneme converters for a Hindi TTS systems. Section 3 describes the modules and algorithms developed at HP Labs India to overcome these problems. Though the effort is focused on Hindi, attention has been paid to make these modules as language independent as possible and scalable to other languages. The following section outlines the reason for choosing Festival as the base for developing such a system.

2. FESTIVAL FRAMEWORK

The Festival framework has been used extensively by the research community in speech synthesis. It has been cho-

sen for implementing the Hindi TTS System because of its flexible & modular architecture, ease of configuration, and the ability to add new external modules. However, TTS system development for a new language requires substantial amount of work, especially in the text processing modules.

The language processing modules in Festival are not adequate for certain languages and the reliance on Scheme as a scripting language makes it difficult for linguists to incorporate the necessary language specific changes within Festival. Thus, we need new tools or modules to be plugged into Festival.

3. TEXT ANALYSIS MODULES CREATED BY HP LABS INDIA

3.1. Grapheme-to-Phoneme (G2P) Conversion

In a TTS system, the G2P module converts the normalized orthographic text input into the underlying linguistic and phonetic representation. G2P conversion, therefore is the most basic step in a TTS system. It is preferable that the phoneme/phone sequence is also appropriately tagged with intonation markups.

Hindi, like most Indian languages, uses a syllabic script that is largely phonetic in nature. Thus, in most cases, there is a one-to-one mapping between graphemes and the corresponding phones. Special ligatures are used to denote nasalization and homo-organic nasals. The Hindi phoneset consists of 86 phones due to the fact that it has a large set of distinct phonemes with a relatively small number of allophones. For example, the stop system in Hindi shows a four way distinction, viz., voiced vs. unvoiced, aspirated vs. unaspirated, for five places of articulation. There is also a phonological distinction between retroflex and alveolar, aspirated and non-aspirated taps. Also, each of the vowels has a phonologically distinct nasalized counterpart. Hindi has a number of borrowed phonemes of Perso-Arabic origin as well, like fricatives /z/ and /f/.

3.1.1. G2P Converter Architecture

As part of the LLSTI initiative [4], a generic G2P conversion system has been developed at HP Labs India. The system consists of a rule processing engine which is language independent. Language specific information is fed into the system in the form of lexicon, rules and mapping. The architecture of the G2P is shown in Fig 1. This G2P framework, has then been customized for Hindi. The default character to phone mapping is defined in the mapping file. The format of the mapping is shown below and explained subsequently.

$$Character\ Type\ Class\ Phoneme$$

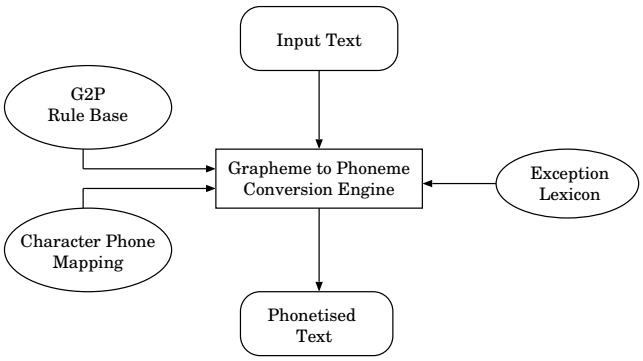

Fig. 1. G2P Converter Architecture

Character	The orthographic representation of the character.
Type	Type of the character, *e.g.* C (for Consonant), V (for Vowel) etc.
Class	The class to which the character belongs. These class labels can be effectively used to write a rule representing a broad set of characters.
Phoneme	The default phonetic representation of the character.

Specific contexts are matched using rules. The system triggers the rule that best fits the current context. The rule format is shown below:

$$\alpha_1\ \alpha_2\ .....\ \alpha_m\ \{\ \beta_1\ \beta_2\ .....\ \beta_n\ \}$$

α_i	Class label of the i^{th} character as defined in the character phone mapping. Together these αs represent the context that is being matched.
β_j	j^{th} action specification node. Each such node has the form:

Action_Type:Pos:Phoneme_Str

Action_Type	This field specifies the type of this action node. Possible values are *K* (Keep), *R* (Replace), *I* (Insert) and *A* (Append).
Pos	The index of the character which is covered by the context of the rule ($1 \leq Pos \leq m$).
Phoneme_Str	Phoneme string used by this action node.

The G2P first looks, for each input word, in the exception lexicon. For the Hindi G2P, this lexicon is the phonetic compound word lexicon which is generated using the algorithm described in section 3.1.4. If the word is present in the

lexicon, the phonetic transcription of the input word is taken from the lexicon itself. Otherwise, the G2P applies the rules on this word and produces the phonetic transcription.

Though Hindi phonology makes it easier to devise letter-to-sound rules, there are certain exceptions to this. Some of these exceptions can be handled by straight-forward rules and some cannot. Schwa deletion is one such problem encountered in Hindi G2P conversion. This problem arises with the phonetization of words containing the schwa vowel (ə). Depending on certain constructs, the schwa vowel is deleted in some cases and retained in others. The orthography, however, provides minimal cues in recognizing these constructs. The following sub-section explains the issue of schwa deletion.

3.1.2. Schwa Deletion

The consonant graphemes in Hindi are associated with an inherent schwa vowel which is not represented in orthography. Let us consider the word *kalam* "pen". The word is represented in orthography by the consonants viz. */k/, /l/* and */m/*. The inherent schwa is associated with each of these consonants and the schwa is pronounced while pronouncing this word. As a rule of thumb in Hindi, a word ending schwa is always deleted. Hence, the final pronunciation of the word is [k ə l ə m]. Vowels other than the schwa are explicitly represented in orthographic text with the help of specific ligatures or *matras* around the consonant.

Let us now take a different word *namkin* "salty". The word consists of four consonants /n/, /m/, /k/, /n/ and one explicitly represented vowel /i/. As stated above, one expects that the schwa will be associated with each of the three consonants. However, in the pronunciation of this word, the schwa after the consonant */m/* is not pronounced. Hence, the correct phonetic transcription of the word is [n ə m k i n].

Several phonological rules have been proposed in the literature [5] for schwa deletion. These rules take into account morpheme-internal as well as across morpheme-boundary information to explain this phenomenon. The Bell Labs Multilingual TTS attempted schwa deletion using morpheme boundary information [6]. The schwa is deleted in the context *VC(C) _ CV* where *V* stands for any vowel and *C* stands for any consonant. Presence of morpheme boundary on the left of the context for the schwa deletion block restricts application of these rules. The schwa is also not deleted if its deletion results in the formation of an invalid consonant cluster.

Encouraging results on schwa deletion using morpheme boundary information is reported in [7]. Morpheme boundary in a word can be detected using a morphological analyzer. However, unavailability of high quality Hindi morphological analyzers prevents adopting this approach.

The following sub-sections describe how the schwa deletion problem has been handled in the Hindi text-to-speech system currently under development at HP Labs India.

3.1.3. Dhvani Schwa Deletion Rules

A set of schwa deletion rules have been incorporated in the Dhvani speech synthesis system [8]. These rules are syllabic and can handle schwa deletion in independent words. Table 1 lists these rules. Here, W(i), C, V, H, Ch and ə stand for the i^{th} character of a word, consonant, vowel, halanth, consonant-halanth combination or a half consonant and schwa vowel, respectively. The rules are applied in a word from right-to-left.

W(n)	W(n-1)	Output
C	" "	C ə
	V	CVC, if W(n-2) is a C; else VC
	C	C ə C
	Ch	C C ə
Ch	" "	C
	V	CVC, if W(n-2) is a C; else VC
	C	C ə C
	Ch	CHCV, if W(n) & W(n+1) are a CV pair & the corresponding 'half' sound is available; else C
V	" "	V
	V	V
	C	CV
	Ch	CV

Table 1. Dhvani Schwa Deletion Rules

A modified version of the schwa deletion rules are used in the Hindi G2P converter. These rules, however, fail in the case of compound words, which are highly prevalent in Hindi. Compound words are formed by two or more words concatenated to form a single word. To illustrate this point, let us consider the words *lok* "public" and *sabhA* "gathering". These two are independent words. However, by combining these two words, the compound word *loksabhA* "lower house of the parliament" is formed. Application of the schwa deletion rules above on this word produces the output [l o k ə s bh A] which is incorrect. The correct phonetic transcription is [l o k s ə bh A].

This problem can be overcome if the constituents of a compound word are analysed separately, which requires a phonetic lexicon of compound words. Such a lexicon is not available for Hindi. Hence, we have proposed an algorithm [9] to automatically extract compound words and identify its constituent parts from a large text corpus.

3.1.4. Compound Word Extraction

The proposed algorithm starts with the assumption that independently occurring words are valid atomic words. A trie-like [10] structure is used to store and efficiently match the words. A *potential* compound word is detected if the currently processed word is part of a word already present in the trie or a word in the trie is a sub-string of the current word. For each word (α), there are several possibilities:

- *Case 1:* α is neither in the trie nor a constituent of any of the words currently in the trie.
- *Case 2:* α is an independent word in the trie.
- *Case 3:* α is the initial part of a compound word currently present in the trie as an independent word.
- *Case 4:* α is the second constituent of a compound word currently present in the trie as an independent word.

In *Case 1*, the algorithm will insert α into the trie. In *Case 2*, no change is needed. In *Case 3*, the algorithm will generate the second constituent (β) for each of the potential compound words where α is the first constituent. After the generation of βs, the end node corresponding to α in the trie becomes a leaf node. The algorithm check for its presence in the trie for each of the generated βs. If β is present in the trie, the algorithm marks the combination (α β) as a compound word. Otherwise, (α β) is inserted into a suspicion list. The algorithm in its current form does not handle *Case 4*.

Before processing each α, the algorithm checks for its presence in the suspicion list. If α is present in the suspicion list as the second constituent, the entry (γ α) is removed from the suspicion list and is marked as a compound word.

To illustrate the algorithm, let us consider a sample lexicon with words in the sequence *lokgAthA, loksabhA, sabhA, lok, gAtha*. If both constituents of a compound are present in the corpus, then the order of reading the compound and its constituent parts is irrelevant. After this the word *lok* is input. The algorithm first checks it in the suspicion list. If it is not found there, the word is matched against the contents of the trie generating the constituent forms *gAthA* and *sabhA*. *sabhA* is already present in the trie but *gAthA* is not yet read in. Hence, *lok sabhA* is marked as a compound word. But since *gAthA* is not present in the trie, *lok gAthA* is inserted into the suspicion list.

After this, the algorithm processes the word *gAthA*. *gAthA* is present in the suspicion list as the second constituent. Hence, *lok gAthA* is removed from the suspicion list and is marked as a compound word. Then the algorithm tries to match *gAthA* in the trie and since it is not present, it is inserted into the trie.

Hence, at the end of the algorithm, the best possible atomic word forms are present in the trie with compounds decomposed into their constituent parts. An improvement of 1.6% in Hindi G2P conversion was observed as a result of incorporating the phonetic compound word lexicon into the Hindi G2P converter. A detailed description of the algorithm along with the issues involved are reported in [9].

3.1.5. G2P Results

Experiments were carried out to test the accuracy of the Hindi G2P. Details of the converter are presented in Table 2.

Total Phones Used	86
Total Character-Phone Mappings Used	62
Total Number of Rules	84
Total Entries in Compound Word Lexicon	48428

Table 2. Customized Hindi G2P Details

To test the phonetisation accuracy of the Hindi G2P, a text file was randomly selected from the Emille corpus [11]. The text was phonetised using the G2P and the phonetic output was manually verified. Phonetisation results are presented in Table 3.

Total Words Tested	3485
Wrong Schwa Deletion	92
Word Phonetisation Accuracy	97.36%

Table 3. Result of Hindi G2P Conversion

3.2. Inventory Design

In concatenative speech synthesis approach, speech is generated by concatenating real or coded speech segments. A recent trend in concatenative synthesis approach is to use large databases of phonetically and prosodically varied speech, thereby minimizing the degradation caused by pitch and time scale modifications to match the target specification and to minimize the concatenation discontinuities at segment boundaries. The quality of output speech primarily depends on the quality of the speech corpus from which the speech segments are obtained. Consequently, the design of the speech corpus is a very important and a critical issue in the data-driven concatenative synthesis approaches. The corpus should contain as much variation as possible both phonetically and prosodically and at the same time the size of the corpus should be as small as possible both to minimize the overhead incurred in database preparation and to facilitate efficient unit selection at synthesis time. In our

work, to generate optimal text to excise the speech corpus, a set of phonetically rich sentences are prepared.

The problem of selecting a set of phonetically rich sentences from a corpus can be mapped to the Set Covering Problem (SCP). A survey of various algorithms for SCP is presented in [12]. The most widely used algorithms for SCP are greedy algorithms [13] [14]. The performances of greedy and spitting algorithms are comparable. We implemented greedy algorithm, since it is marginally better and less time consuming [15].

This algorithm builds the cover step-by-step. The algorithm recursively selects sentences. The first sentence is the sentence with the largest unit count. All units present in the sentence are then removed from the to-be-covered list of units. The sentence which covers maximum number of remaining units is selected next. This process continues until all the units are covered or the corpus is exhausted.

The units to be covered can be assigned weights based on their frequency in a corpus. If complete coverage is possible, then the weight of a unit can be made equal to the inverse of its frequency. This makes the algorithm focus more on less frequent units. The idea is that the high frequency units are anyway covered en-route. However, if it is known *apriori* that complete coverage is not possible, then the weight of the unit can be made equal to its frequency.

The Hindi G2P converter was used to phonetize the corpus. A diphone frequency list was prepared by running CMU-Statistical Language Modeling Toolkit [16] on the phonetised text corpus. An exhaustive list of 7392 diphones was generated using the Hindi phoneset. Phonotactic constraints of the language were not taken into consideration while generating the diphone list. Word beginning, word ending, sentence beginning & sentence ending contexts were considered while selecting the sentences. Silence-phone combinations were also taken into account. Text selection was carried out in three phases. Details of various corpora used in the text selection are presented in Table 4.

3.2.1. Phase 1 & 2

Webdunia Newspaper text, which is part of the Emille corpus, was used in this step. From a total of 163,517 sentences, 665 sentences were selected resulting in the coverage of 2404 diphones.

In the second phase, the idea was to cover the units not covered in the first phase. *Ranchi Express* newspaper text was used in this phase. From a total of 142122 sentences, 303 sentences were selected covering 379 diphones. Hence, total diphone coverage at the end of this step was 2783.

3.2.2. Phase 3

At the start of this phase, the three remaining corpora *viz. Indianinfo*, *Miscellaneous (Misc.)* and *Spoken* were analysed to see their coverage of the diphones which were not covered after the first two phases. The results are shown in Fig 2. It is clear from the graph that *Misc* corpus had the highest coverage. The *Spoken* corpus showed great promise in terms of coverage but the limited size of this corpus restricted its usage at this stage. Hence, the *Misc* corpus was used for text selection at this stage. Another interesting observation was the formation of the plateau with the addition of more newspaper text which is of the same genre as that of text used in Phases 1 & 2. This illustrates the fact that phonetic coverage attains saturation once a significant amount of text of one type is analysed and additional text of the same type is unlikely to give significant increase in coverage.

Corpus Name	Genre	Total Sentences	Total Words
Webdunia	Newspaper	163175	3393192
Ranchi Exp.	Newspaper	142122	2910613
Indiainfo	Newspaper	36057	607017
Misc.	Assorted	128537	1728644
Spoken	Radio Text	17720	507952

Table 4. Hindi Corpora Statistics

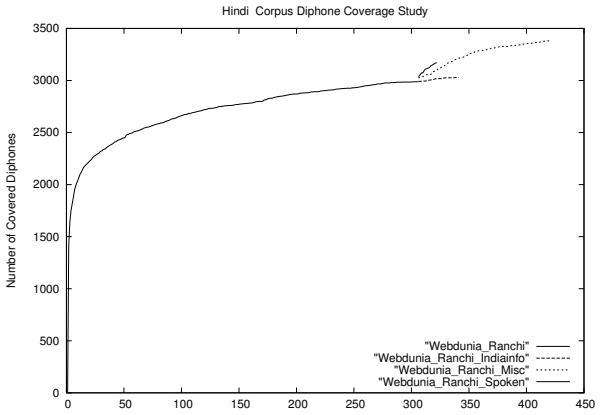

Fig. 2. Hindi Corpora Diphone Coverage Study

Another aspect observed during the analysis at this phase was that the covered-diphone to selected-sentence ratio was nearing one *i.e.* a complete sentence was being selected to cover just one unit. This is undesirable since the size of the unit inventory is directly proportional to the number of selected sentences and an unnecessarily big speech corpus results in additional overhead in speech-segmentation and increased search time in the unit selection stage. Hence, instead of the complete sentence, just the word containing the particular diphone was included in the cover. This strategy selected 327 unique words resulting in a coverage of 373

diphones.

Total Initial Diphones	7392
Total Sentences Analysed	433834
Count of Selected Sentences	968
Count of Selected Words (Phase 3)	327
Total Diphones Covered	3156
Total Uncovered Diphones Count	4236

Table 5. Results of Hindi Text Selection

Selected sentences were hand corrected to remove typographical errors and spelling mistakes. Final coverage results are presented in Table 5. The selected sentences and words are just **0.3%** of the three text corpora (*Webdunia, Ranchi Exp., Misc.*) which were used in text selection.

As a by product, the final uncovered diphone list can be considered as phonotactically invalid Hindi diphones since these diphones remained uncovered after analysing a large text segment of more than 400,000 lines. A separate list of words was prepared to cover the consonant clusters in Hindi.

4. CONCLUSION

An effective G2P conversion module has been developed for Hindi. This combines our new algorithm for automatically creating a compound word lexicon from a text corpus with the Dhvani schwa deletion rule base. An inventory of phonetically rich Hindi sentences has also been created using a greedy algorithm on Hindi text corpora. By using a large text corpora from various sources, we are able to extract the phonotactic constraints in Hindi by identifying the forbidden diphones.

5. ACKNOWLEDGEMENT

The authors would like to thank Dr. K.S.R. Anjaneyulu (HP Labs India) for his guidance and inputs through out the course of this work. Special thanks to Deepa S.R. (BITS, Pilani) for her contributions in the compound word extraction.

6. REFERENCES

[1] S.P. Kishore, Rohit Kumar, and Rajeev Sangal, "A Data-Driven Synthesis approach for Indian Languages using Syllable as Basic Unit," in *Intl. Conf. on Natural Language Processing (ICON)*, 2002, pp. 311–316.

[2] J. Rama, A.G. Ramakrishnan, and R. Muralishankar, "A Complete TTS system in Tamil," in *IEEE Workshop on Speech Synthesis*, 2002.

[3] A. Black and P. Taylor, "Festival speech synthesis system: system documentation (1.1.1)," Tech. Rep. HCRC/TR-83, Human Communication Research Centre, 1997.

[4] Roger Tucker, "Local language speech technology initiative," (http://www.llsti.org), 2003.

[5] Manjari Ohala, *Aspects of Hindi Phonology*, Motilal Banarsidas, Delhi, 1983, First Printing.

[6] B. Narasimhan, R. Sproat, and G. Kiraz, "Schwa deletion in hindi text-to-speech synthesis," in *Workshop on Computational Linguistics in South Asian Languages*, Oct. 2001.

[7] M. Choudhury and A. Basu, "A Rule-based schwa deletion algorithm for Hindi," in *Proc. International Conference On Knowledge-Based Computer Systems*, Navi Mumbai, 2002, pp. 343 – 353.

[8] "Dhvani - TTS System for Indian Languages," (http://dhvani.sourceforge.net), 2001.

[9] Deepa S.R., A.G. Ramakrishnan, Kalika Bali, and Partha Pratim Talukdar, "Automatic Generation of Compound Words Lexicon for Hindi Speech Synthesis," *Language Resources & Evaluation Conference (LREC)*, 2004.

[10] Jeffrey D. Ullman Alfred V. Aho, John E. Hopcroft, *Data Structure and Algorithms*, Addison-Wesley Publishing Company, Reading, MA, 1983.

[11] "Emille corpus," (http://www.emille.lancs.ac.uk), 2003.

[12] A. Caprara, M. Fischetti, and P. Toth, "Algorithms for the Set Covering Problem," Tech. Rep. No. OR-98-3, DEIS-Operations Research Group, 1998.

[13] Jan P. H. van Santen and Adam L. Buchsbaum, "Methods for optimal text selection," in *Proc. Eurospeech '97*, Rhodes, Greece, 1997, pp. 553–556.

[14] Ronald L. Rivest Thomas H. Cormen, Charles E. Leiserson, *Introduction to Algorithms*, Prentice-Hall of India, New Delhi, 2000, Second Printing.

[15] H. Franois and O. Boffard, "The Greedy Algorithm and its Application to the Construction of a Continuous Speech Database," in *3rd International Conference on Language Resources and Evaluation (LREC 2002)*, 2002, vol. 5, pp. 1420–1426.

[16] Ronald Rosenfeld, "The CMU Statistical Language Modeling Toolkit, and its use in the 1994 ARPA CSR Evaluation," in *ARPA Spoken Language Technology Workshop*, January 1995.