System for Identifying and Correcting Invalid Words in the Devanagari Script for Text to Speech Engine

Damodar Magdum, Maloji Suman

Abstract— The Text To Speech (TTS) system takes text as an input and generates speech as an output. If input text is incorrect then overall quality of speech output may degrade. The main aim of the proposed system is to provide correct input text to the TTS. The system takes Unicode word as an input, identifies invalid word and corrects it by inserting, deleting or updating characters of the word. In this system, the State Machine is used to identify and correct invalid word in the Devanagari script which in turn is based on rules. Rules are developed for converting character to input symbol. Actions and States are identified for State Machine. Finally, the state transition table is developed for validation and correction of word. Using this system, incorrect words of the Devanagari script can be corrected to valid words (word contains all the valid Devanagari syllables) based on Devanagari script grammar. Since, all Devanagari characters are not present in Hindi language; this system will correct these non-Hindi characters to Hindi.

Index Terms— Text to Speech, Unicode words, State Machine, State transition table, Devanagari Script.

I. INTRODUCTION

There are many reasons for invalid text data in the Indian languages. The major reasons for invalid text data are complex structure of Indian languages. Unlike English language Hindi language have more character sets like Consonants, Vowels, Vowel signs or Matras, Vowel Modifier, Halant and Nukta etc.

In addition to the above character sets, The Indian languages are syllabic in nature and there are strict rule for syllable formation [1]. As there are strict rule for writing word, there may be chance of invalid word and we need to correct it before it is used in any application. The main aim of proposed system is to correct Hindi language text and as

Hindi is written using Devanagari script, the proposed system indentifies invalid syllable in Devanagari script and correct it which can be further used in Text-to-Speech (TTS) system [10] [11]. Lack of good inputting device specific to Indian language. The total number of characters in Indian languages is more than English and it is very difficult to provide specific key for individual character so there may be chance of input error due to lack good inputting method using keyboard which is initially developed for English. Words borrowed from English or Sanskrit is difficult to write by Hindi users. As many user may not know how to write the word so there may be chance of error. Lack of knowledge in terms of suing different types of keyboards for inputting like inscript, phonetic or type writer. As each keyboard required different character order for inputting words for Indian language [5].

Revised Manuscript Received on April 12, 2019.

Damodar Magdum, Research Scholar, Koneru Lakshmaiah Education Foundation, Vaddeswaram (A.P.,INDIA). (E-mail: damodarm@cdac.in)

Dr. Maloji Suman, Professor, Koneru Lakshmaiah Education Foundation, Vaddeswaram (A.P., INDIA). (E-mail: suman.maloji@cdac.in)

II. IMPORTANCE OF TEXT CORRECTION IN THE TTS ENGINE

The TTS engine takes text as input and generates speech output. There are mainly two engines present namely linguistic engine and acoustic engine. Linguistic engine takes input from front end and acts as a pre-processor for the acoustic engine which actually does synthesis and generates speech output. Linguistic engine converts input text to phonetic string that is used by acoustic engine for further processing. Text cleaning, segmentation, text normalization, grapheme to phoneme and schwa deletion are the main module of Linguistic engine. Quality of acoustic engine depends upon accuracy of linguistic engine modules [3]. Figure 1 shows Text-to-Speech block diagram with input and output of each modules.

As shown in Figure 1, the input text is invalid as there are Matra following Matra which forms invalid word as per ISCII rules [1].

And the invalid कििताब word is then corrected using Text Cleaning module. But instead of correcting invalid word, if system passes wrong words to the Grapheme to Phoneme module, then it produces wrong phonetic mapping.

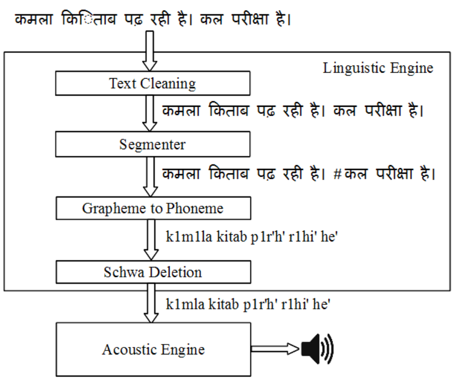
कमला कििताब पढ़ रही है। कल परीक्षा है।
Linguistic Engine
Text Cleaning
कमला किताब पढ़ रही है। कल परीक्षा है।
Segmenter
कमला किताब पढ़ रही है। #कल परीक्षा है।
Grapheme to Phoneme
k1m1la kitab p1r'h' r1hi' he'
Schwa Deletion
k1mla kitab p1r'h' r1hi' he'
Acoustic Engine

Fig. 1. Text-to-Speech Block Diagram

For example. After entering word like कििताब, phonetic mapping is generated as k+i+i+t+a+b. And this phonetic string is passed to the acoustic engine which converts speech output with two ि matras in the output which degrades the speech quality. Instead of passing कििताब to Grapheme to

Phoneme module, the word is corrected as किताब and then sending it to Grapheme to Phoneme module, it generates correct phonetic as k+i+t+a+b. So the speech output is also generated correct which improves the speech quality.

III. LITERATURE REVIEW

The Hindi script grammar [1] defines characters in sub-set as consonants, vowels, and matras, Vowel Modifiers, Halant and Nukta for the Hindi language. It defines type of syllables with vowel as a node and consonant as a node in Hindi. It also defines list of punctuations used with Hindi and define rule for use of Anuswar and Chandrabindu with Vowel and Matra.

ISCII 91[1] provides information about rule for ordering characters in a given syllable. There are strict rule for which sub-set is precede and follow which sub-set. It is defined by Backus-Naur Formalism (BNF).

Hindi Unicode chart [12] helps in making final rules for system as it gives proper indexing of each character. We can exploit its structure and ordering for the other Indian languages also. After making rules for valid syllable or words we required to parse the input words.

Bhalerao and Satpute [2] have not provided any correction method if syllable is invalid, it only displays syllable is invalid to the application. But there are applications that require text correction automatically like the TTS system. They have only provided invalid syllable in terms of the Devanagari script but some applications are not script specific; they are language specific like the TTS system. The TTS system is for Hindi language and not for the Devanagari script. So there are other corrections required before sending it to the system as an input.

Example: अ+छ+्+च+ा = अछ्चा is valid as per work shown by Bhalerao and Satpute [2], as अछ्चा is valid Devanagari word. But it invalid in terms of the TTS system as aspirated character छ is preceded un-aspirated character च. So the system should be required to correct अछ्चा word and produce अ+च+्+छ+ा = अच्छा as an output [6].

There are many other words those are valid as per ISCII-91 Devanagari script, but required little correction to use it with the TTS as TTS required words which are phonetically correct.

IV. STATE MACHINE

Study of the Finite State Machine and Turing Machine [13] [14] is done for implementation of language rules to indentify invalid word. Here the system process one character at a time and check whether character is correct or not and if it is invalid then correct using specified action.

For designing state machine first step is to identifying input symbols, states and action taken on each state.

IV.I Input symbol

There are many characters in Indian language so characters should be grouped based on each on the similarity and each group maps with the specific input symbol. Table 1 shows the Hindi characters and grouped into specific category and mapped with input symbols [7].

These groups are further divided into sub groups to handle specific rules like Consonants are divided for handling Nukta consonants, "ज्ञ" and specific nasal groups based on Vargas; Vowels and Matras are divided based on Shirorekha means if Matra or vowel written above Shirorekha or not; Pure nasal divided to handle "ज्ञ" [8].

IV.II State

There are 9 states in the system and details of each state are given in the Table 2.

Table 1. Character to Input Symbol Mapping

Sr. No	Input Symbol	Mapped Characters	Description
1	C	क, ख, ग, घ, च, छ, ज, झ, ट, ठ, ड, ढ, ण, त, थ, द, ध, न, ऩ, प, फ, ब, भ, म, य, र, ऱ, ल, ळ, ऴ, व, श, ष, स, ह, क़, ख़, ग़, ज़, ड़, ढ़, फ़, य़	Consonant
2	V	अ, आ, इ, ई, उ, ऊ, ऋ, ऌ, ऍ, ऎ, ए, ऐ, ऑ, ऒ, ओ, औ, ॠ, ॡ	Vowel
3	M	ा, ि, ी, ु, ू, ृ, ॄ, ॅ, ॆ, े, ै, ॉ, ॊ, ो, ौ, ॢ, ॣ	Matra (Vowel sign)
4	VM	ँ, ं, ः	Vowel Modifier
5	NASAL	ङ, ञ	Pure Nasal
6	NUKTA	़	Nukta
7	HALANT	्	Halant
8	DELIMITER	ॐ, ।, ॥, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9	Special character, Punctuation, Numbers, etc.
9	EOW	NULL	NULL character
10	JUNK	All other characters	All other characters

Table 2. States of the system

Sr. No.	States	Description
1	ST_START	The system starts with this state.
2	ST_C	When input symbol is C then system comes into this state

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

3	ST_V	When input symbol is V then system comes into this state
4	ST_M	When input symbol is M then system comes into this state
5	ST_VM	When input symbol is VM then system comes into this state
6	ST_NASAL	When input symbol is NASAL then system comes into this state
7	ST_HALANT	When input symbol is HALANT then system comes into this state
8	ST_DELIMITER	When input symbol is DELIMITER then system comes into this state
9	ST_END	When input symbol is EOW then system comes into this state

IV.III Action associated with current state and symbol

The action is taken based on current input symbol and current state. All the possible actions are listed in Table 3.

Table 3. Possible action

Sr. No	Action Name	Action Description
0	TM_NULL	No Operation just modify states
1	TM_del_curr	Delete current character
2	TM_isNukta	Check if previous consonant is Nukta or not and if it is not Nukta character delete Nukta
3	TM_combine_V_M	combine V-M and M-M to form new vowel or Matra
4	TM_swap	swap current and previous characters
5	TM_insert_prev	insert Halant
6	TM_replace_prev	replace nasal or Chandrabindu by Anuswar
7	TM_del_prev	Delete previous character
8	TM_isValidCC	check if there are valid CC cluster
9	TM_replace_curr	replace current character
10	TM_handle_gy	handle ज्ञ

IV.IV State transition table (STT)

The STT combines input symbols, states and action taken on current input symbol and current state. Which action should be taken is defined in the STT so there are two STT. Table 4 shows State Transition Table for correction and Table 5 shows State Transition Table for deletion. Table 6 shows notations used to state machine examples.

Table 4. State Transition Table for Correction

States / Input Symbol	C	V	M	VM	NASAL	NUKTA	HALANT	DELIMITER	JUNK	NULL
ST_START	0	0	1	1	1	1	1	0	1	0
ST_C	0	0	0	0	0	2	0	0	1	0
ST_V	0	0	3	9	0	1	1	0	1	0
ST_M	0	0	3	9	0	4	1	0	1	0
ST_VM	0	0	4	1	1	4	1	0	1	0
ST_NASAL	5	6	4	1	1	4	0	0	1	6
ST_HALANT	8	7	7	7	10	4	1	0	1	0
ST_DELIMITER	0	0	1	1	1	1	1	0	1	0

Table 5. State Transition Table for deletion

States / Input Symbol	C	V	M	VM	NASAL	NUKTA	HALANT	DELIMITER	JUNK	NULL
ST_START	0	0	1	1	1	1	1	0	1	0
ST_C	0	0	0	0	0	2	0	0	1	0
ST_V	0	0	1	0	0	1	1	0	1	0
ST_M	0	0	1	0	0	1	1	0	1	0
ST_VM	0	0	1	1	1	1	1	0	1	0
ST_NASAL	0	0	1	1	1	1	0	0	1	0
ST_HALANT	0	1	1	1	10	1	1	0	1	0
ST_DELIMITER	0	0	1	1	1	1	1	0	1	0

Table 6. Notation used in State Machine Example

Notation	Description
Dir	Direction
NS	Next State
I/P	Input
O/P	Output
L	Move cursor to left by one character
R	Move cursor to right by one character
N	Not move
D	Go to start of string (index=0)

IV.V State Machine Example

The word “कििताब” is broken as “क+ि+ि+त+ा+ब”. The step by step process of state machine for word “क+ि+ि+त+ा+ब” is given in the Table 7. The system starts with ST_START state. The input character “क” is mapped to the symbol C. Based on current state and symbol (ST_START, C), TM_NULL action is executed. It calculates next state as ST_C and move cursor to the R (Right by one character). Same way, the system scan each character and produce the correct Devanagari word [4] [5].

Figure 2 shows state machine diagram of word “कििताब”.

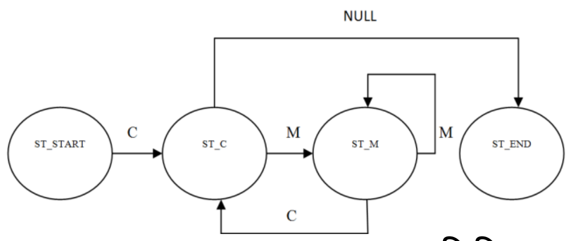

Fig. 2. State Machine Example word कििताब

Table 7. State Machine Example

Sr. No.	Current State	Input character	Current Symbol	Action	Next State	Direction
1	ST_START	क	C	TM_NULL	ST_C	R
2	ST_C	ि	M	TM_NULL	ST_M	R
3	ST_M	ि	M	TM_del_curr	ST_M	N
4	ST_M	त	C	TM_NULL	ST_C	R
5	ST_C	ा	M	TM_NULL	ST_M	R
6	ST_M	ब	C	TM_NULL	ST_C	R
7	ST_C	NULL	NULL	TM_NULL	ST_END	N

V. SYSTEM DESIGN

Figure 3 shows flow chart of proposed state diagram. There are mainly four modules in system. Initialization module load all rules required for the system and Un-initialization module free all the loaded resources. Both the modules executed once for application lifetime. Script based validation and Language Specific Mapper (LSM) executed for each input word.

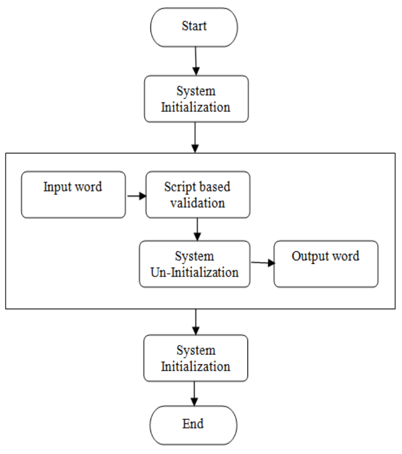

Fig. 3. Flow chart of state

V. I Initialization Module

This module takes language mnemonic as an input which is used for calculate language offset. Language offset is used for loading language rules. This language map rules loaded before system will perform validation. Other three rule files like vowels and matra lookup, Devanagari to Hindi mapping lookup, invalid consonant-consonant lookup are optional and load only if further correction is required. It calculates language offset from language mnemonic given as an input to the module. If language mnemonic is not provided in this module then system reports warning to the application.

V. II Script Based Validation

The aim of the module is to identify the invalid words of the Devanagari script and correct it based on the State Transition Table (STT) provided to the module. The system takes word to be validated and flag to tell the module which STT to be used for correction as an input. The system has two transition tables. First transition table contains the actions that will correct the invalid words and second table just delete the current invalid character of the word to correct the input word. The system always start with the initial state called ST_START. The system scans the input word character by character and map the current input character to the input symbols from the State Transition Table. The action is executed and perform task associated with the given action. The action is executed and perform task associated with the given action. The details of the task performed by action are given in to the table 3. Each action performs its task and also calculates next state of the machine, and direction in which cursor moves. This loop continues till the end of word and after the loop word is corrected if it is invalid and module also provide feedback for word is corrected or it is already valid word [8].

V. III Language Specific Mapper (LSM)

The Language Specific Mapper (LSM) is mainly correct Devanagari characters those are not present in the Hindi language using the lookup table [15]. The module takes corrected word from the Script based validation module as an input and generates word as an output with all the characters present in the Hindi language [9].The TTS system map the grapheme (written symbol or character) to the phoneme and this Grapheme to Phoneme module is language specific so it required word with all the characters of that language only. So the LSM maps all the character to the Hindi specific characters.

V. IV Un-Initialization Module

This module release all the resource acquired by the system. The main aim of the module is to free all the loaded rules and lookup tables. The system can be used for Hindi and other languages for word correction. To use the system for other language, the system should release previous language rules. So this module release rules used by the system for current language.

VI. RESULT

The system result shown in Table 8. The first output is obtained by deleting current invalid character, while second output is obtained by intelligently inserting, updating, or deleting characters in word and after the correction word is further processed by the Language Specific Mapper (LSM) module. This LSM module has been tested over 6000 high frequency words. We received accuracy result up to 80%

Table 8. Result of the system for deletion and correction of the words

Input word	Delete the current invalid character	Word Correction
ऄेक्षय	ऄक्षय	अक्षय
ऄौमकार	ऄमकार	ओमकार
अछचा	अछचा	अच्छा
आङ्ख	आङ्ख	आङ्ख
आ◌ेयल	आयल	ओयल
आ्या	आया	आया
ईँट	ईँट	ईंट
कं◌ँत	कंत	कंत
कंङत	कंत	कंत
कं◌्सव	कंसव	कंसव
कङँत	कङत	कन्त
कङङत	कङत	कन्त
काङ्त	काङ्त	कान्त
का◌ेयल	कायल	कोयल
कुञआ	कुञआ	कुंआ
पं◌िक	पंक	पिंक
पङिक	पङक	पिङ्क
प़फ	पफ	पफ
पिँक	पिँक	पिंक
पिं◌़क	पिंक	पिंक
पिङ्क	पिङ्क	पिङ्क
रघ्ंविर	रघ्विर	रघंविर
रघ्उविर	रघ्विर	रघउविर
रघ्◌ुविर	रघ्विर	रघुविर
विप्ञान	विप्न	विपान
विप्ञानी	विप्नी	विपानी
वि◌िज्ञानी	विज्ञानी	विज्ञानी
हँस	हँस	हँस
◌़खाब	खाब	खाब
◌िकताब	कताब	कताब
◌्सव	सव	सव
ॐकार	ॐकार	ॐकार

VII. DISCUSSION

The proposed system provides corrected output which is more preferred by the text to speech (TTS) system instead of only deleting the current invalid character. The system corrects the word with valid syllable structure as the word is wrong phonetically. In the Indian language word should not contain aspirated character before un-aspirated as Consonant- Halant - Consonant cluster. So the proposed system correct the order of the consonants.

The system does not only delete the current invalid character, but insert, update, swap, delete previous character. The system corrects word with combining vowel and matra to form new vowel, swapping consonant to correct the Consonant - Halant - Consonant cluster. The Language Specific Mapper (LSM) module replaces characters which are not present in the Hindi but available in the Devanagari.

VIII. CONCLUSION

The proposed system used to identify invalid words and correct it for the use as an input to the TTS system. To accomplish this goal, language specific rule is determined. This extended rules were used to identify and correct the invalid words for specific requirement of the text to speech system. The extended rules were the major requirement to build the system. The rules were implemented using the state machine into the system. The state machine is finalize after mapping of character to symbol and identification of states and required actions.

IX. ACKNOWLEDGEMENT

The author is thankful Mr. Ronak Shah, Principal Technical Officer, C-DAC, Pune and GIST, TTS Team for their support to this work.

REFERENCES

1. BIS.INDIAN SCRIPT CODE FOR INFORMATION INTERCHANGE (ISCII), 1991.
2. Bhalerao R,Satput P, Mechnism for identifying invalid syllables in devnagari script, United States Patent Application Publication,2011
3. Text to Speech Testing Strategy Version 2.1, Technology Development for Indian Languages Programme(Govt of India), 2004.
4. Bisht R,“A Survey of Applications of Finite Automata in Natural Language Processing,” International Journal on Emerging Technologies, 2017, 62-64.
5. Rahul S, Soma P,“A rule based approach for automatic clause boundary detection and classification in Hindi,”2014.
6. Omkar K,“Modern Hindi Grammar. Indian Institute of Language Studies,” 2009.
7. Script Grammer for Hindi Language Technology Development for Indian Languages (TDIL) Programme, Goverment of India, 2019.
8. Anil Kumar S,“A computational phonetic model for Indian language scripts. In Constraints on Spelling Changes,” Fifth International Workshop on Writing Systems, 2006.
9. Anand R,Tanuja S, Sathish P, Santhosh Y, Mohit B, Kishore.P, Alan B,“Text Processing for Text-to-Speech Systems in Indian Languages,” Proceedings of 6th ISCA Speech Synthesis Workshop SSW6, Bonn, Germany, 2007.
10. https://en.wikipedia.org/wiki/Speech_synthesis
11. https://www.cdac.in/index.aspx?id=mlc_gist_speechtech, 2019
12. http://www.unicode.org/charts/PDF/U0900.pdf, 2019.
13. https://en.wikipedia.org/wiki/Turing_machine
14. https://en.wikipedia.org/wiki/Finite-state_machine
15. https://techwelkin.com/tools/character-map/