LTS using Decision Forest of Regression Trees and Neural Networks

Tanuja Sarkar, Sachin Joshi, Sathish Chandra Pammi[1]
Kishore Prahallad[2]

[1]International Institute of Information Technology, Hyderabad, India.
[2]Language Technologies Institute, Carnegie Mellon University, USA.

tanuja@students.iiit.ac.in, sachin_sj@students.iiit.net,
sathishp@students.iiit.net, skishore@cs.cmu.edu

Abstract

Letter-to-sound (LTS) rules play a vital role in building a speech synthesis system. In this paper, we apply various Machine Learning approaches like Classification and Regression Trees (CART), Decision Forest, forest of Artificial Neural Network (ANN) and Auto Associative Neural Networks (AANN) for LTS rules. We used these techniques mainly for Schwa deletion in Hindi. We empirically show that the LTS using Decision Forest and Forest of ANNs outperforms the previous CART and normal ANN approaches respectively, and the non discriminative learning technique of AANN could not capture the LTS rules as efficiently as discriminative techniques. We explore use of syllabic features, namely, syllabic structure, onset of the syllable, number of syllables and place of Schwa along with primary contextual features. The results showed that use of these features leads to good performance. The Decision Forest and forest of ANNs approaches yielded phone accuracy of 92.86% and 93.18% respectively using the newly incorporated features for Hindi LTS.

Index Terms:letter, decision trees, decision forest, Artificial Neural Network, Auto Associative, CART, features, Schwa

1. Introduction

Speech synthesizers [1], for a natural language, comprises of two kinds of processing modules, text processing and wave generation. In this paper, we have focused on , Letter-to-Sound rules (LTS) module, which is an important part of text processing module in speech synthesizer. Our work deals with LTS for Hindi and can also be applied to all other Indian languages, since they are phonetically similar to each other. LTS module maps a particular letter of the language to its corresponding sound representation, given a definite context.

An $akshara$ in Indian language scripts is a syllable and is typically of the form: $C*VC*$, where C is a consonant and V is a vowel. The subunits of $akshara$ which are used to represent these consonants and vowels are called letters. There is a fairly good correspondence between what is written and what is spoken. Indic languages are syllabic in nature, with an important concept of the inherent vowel Schwa (/a/). Unless otherwise indicated, each consonant is typically followed by this vowel sound /a/. Indian languages require rules for letter to sound mapping, which varies from one language to another. The rules can be either written manually or generated by a machine learning algorithm. Quite often manually written rules give quite fair results [2]. In some cases where there are more exceptions to a rule, the results of the hand-written rules are poor [3]. It has been generally observed that data-driven approaches give better results than the rule based, for such languages [4]. Moreover, in rule-based systems, a set of hand-written rules is provided with the input data for each language. For these systems, it is necessary to provide a separate rule-set for every new language. It may be possible that the rules in the rule-set may not be written with full precision. This makes LTS module less robust.

Problem of Schwa Deletion: There are various challenges involved in building the LTS module for Indian languages. Most of the Indian languages, face the problem of pronunciation of the inherent vowel called Schwa. In a word, the Schwa is either deleted, or not deleted or replaced depending on the context. These contextual rules are either handwritten for rule-based system or generated by the decision trees for data-driven systems. For example, the following words written in ITrans-3 [5] notation describes the Schwa in Hindi,
/matagandãnaa/ — /mat#gandãnaa/ : Here '#' represents the deleted Schwa.
/aaveidanapatroon'/ —- /aaveidan#patroon'/. To illustrate how improper Schwa deletion can render the speech incomprehensible, compare the word "dhad-akanein$^{\prime}$'(pronounced as "dhad-kanein$^{\prime}$') [noun. heart-beats] with "dhad-akanei"(pronounced as "dhad-aknei") [verb. To beat (heart)], where Schwa following 'k' is deleted. Without any Schwa deletion, not only will the two words sound very unnatural, but it will also be extremely difficult for the listener to distinguish between the two.

Until recently , for Indian languages, most of the LTS modules were modelled using rule-based approach. It has been observed that machine learning (ML) can be used to built robust LTS modules. In this paper, we propose to apply ML concepts like CART, Decision Forests, ANN and AANN to LTS problem in Indian languages. Here, we have tried to analyse and compare performances of these machine learning techniques to come up with better system. The rest of the paper is organized as follows. Section 2 discusses the previous approaches for building letter-to-sound module. Section 3 describes building the database for the LTS module and feature extraction. Section 4 talks about LTS using CART. Section 5 discusses the LTS system build with AANN and Decision Forest. Section 6 analyses the results of the various experiments with various data-driven approaches with Hindi language. Section 7 concludes the whole paper with basic points highlighted briefly.

2. Related Work

LTS module is generally modelled using rule-based and statistical methods. Both the methods have some advantages and disadvantages which make them usable in some specific areas.

Table 1: *Detailed description of the Hindi Database*

Number of syllables	Number of words
1	160
2	2872
3	6987
4	5391
More than 5	3496
	18906

Rule-based: Traditional models for LTS rules have been built using rule-based approaches. The main advantage of these systems is that they do not require a huge training corpus. Rather a small list of rules is provided as the core of the system. Rule based approaches for LTS in Indian languages were reported in [2]. This paper discussed the problem of Schwa in the context of morpheme boundary. But rule-based systems lack robustness as it is difficult to write complex rules applicable to all conditions. For a new language, a new rule-based system is to be built with a new set of learning-rules, which is time comsuming.

Statistical Methods: To improve the robustness of the rule-based system, statistical systems were introduced [6],[7]. Various statistical techniques like decision trees, neural networks [8] etc. were experimented on. Paper [9], discusses the problems of building a LTS module, for English, using CART. Letter to sound was also built using ANN [10]. Selection of proper features plays an important role in building a statistical LTS system [11]. Generally the statistical models depend on the features that can be easily extracted from the data.

3. Data Preparation

The text corpus collected to build Hindi TTS, contains 12,230 sentences, which have 18,906 unique words. These sentences were recorded by professional native Hindi speaker. Corresponding pronunciation for all these unique words were written by native Hindi speaker with the help of recorded speech utterances. Data corpus used for LTS was statically balanced as shown in Table 1. It can be observed from the above table that, database for Hindi contains 18,906 unique words. Among these, 3,782 (20%) words were selected by sampling one word in every 5 words, for testing and remaining 15,124 words used for training.

Feature extraction is fundamental to any machine learning technique. Features contain the information for uniquely classifying a given unit. The experiments were done with incremental procedure of widening the feature vector length. Primarily previous and next context letters were chosen as basic features. It is well known fact that Schwa deletion in Indian languages is affected by the position of the Schwa in the word. Schwa in the first syllable is always retained regardless of the contextual features. Schwa deletion in other syllables of a word varies depending on the context. The structure of Schwa containing syllable like CV, CCV or CVC is also a promising feature for Indian languages. According to [2], for any conjugate syllable or cluster of consonants that ends in y, r, l or v, the Schwa following the cluster is retained. So other features like syllable position, structure, onset etc. were added to primary features. Here we exemplify the types of feature sets with example of fourth phoneme (Schwa) occuring in word $matagand - anaa$.

- 2 Level contextual features like previous and next letters: a t g a a
- 3 Level contextual features like previous and next letters: a t g a a m nd−
- 4 Level contextual features like previous and next letters: a t g a a m nd− $ a
- 4 Level contextual + syllable structure: a t g a a m nd− $ a CVC
- 4 Level contextual + syllable structure + onset of the current syllable: a t g a a m nd− $ a CVC m
- 4 Level contextual + syllable structure + onset of the current syllable + place of Schwa (first, middle or end syllable): a t g a a m nd− $ a CVC m 1
- 4 Level contextual + syllable structure + onset of the current syllable + place of Schwa + number of syllables in the word: a t g a a m nd− $ a CVC m 1 3

After feature extraction, training data consist of 22973 vectors and test data has 9845 feature vectors. This data was used for the experiments with CART and AANN.

4. LTS Using CART

CART [6],[7] is a data driven technique based on successive spliting of nodes into relatively homogeneous child nodes. The basic CART building algorithm is a greedy algorithm in that it chooses the locally best discriminatory feature at each stage in the process. The stop parameter specifies the minimum number of samples necessary in the training set before a decision of split is made. Normally the smaller the stop value the more over-trained the models may become. Determining the optimal stop value for Hindi LTS is the key issue. We performed various experiments to conclude the suitable stop value.

5. LTS Using Artificial Neural Networks

Artificial Neural Networks is an information processing paradigm inspired by biological Nervous System. ANN contains large number of interconnected units called Neurons working in unison which make them a robust technique [12].

Input Output Representation: Input data normalization is key factor which governs performance of ANN to large extent. We found that ANNs work best with binary input values. So all the feature vectors were represented in binary string format. The 0's in binary strings were represented by -1 to make the inputs perfectly normalized. The output of ANN was only one value, 1 for Schwa deletion and -1 for retaining it.

Network Parameters and Training: We used three layered feed forward ANN and tested it for multiple configurations. Number of neurons in hidden layer is a key parameter to be determined. We found that number of hidden layer Neurons being equal to number of inputs is the optimal configuration. The activation function of hidden and output layer neurons was tangent sigmoid. Network was trained using back propagation algorithm with learning rate of 0.0001. It was trained for 100 iterations.

Decision Forest: A decision forest is a set of several decision trees [13]. These trees can be formed by various methods or by single method, but with different parameters. The building procedure uses different sub-samples of observations having different characteristics over one and the same phenomenon. Such many-sided consideration of a problem, as a rule, gives the improvement of quality of forecasting and a better understanding of laws of the researched phenomenon. The decision

forest algorithm uses a stop value of 1 while building the individual decision trees. In this work, we have used concept of "Bagging" [14] to build decision forest of total 11 trees and used voting scheme to predict output.

Bagging: Bagging (Bootstrap AGGregatING) produces replications of the training set by sampling with replacement. Each replication of the training set has the same size as the original set, but some examples can appear more than once while others do not appear at all. Bagging should only be used if the learning machine is unstable. Bagging improves the estimate if the learning algorithm is unstable and reduces the variance of predictions without changing the bias. We created total 11 bags of data.

Voting Scheme: Once the decision forest is built, the member trees are then used with the test data set for predicting the output of each of the vectors of the test set. Then the final output is chosen based on voting strategy, which will weigh the most popular output from the set of outputs. Given the several outputs, a weightage-by-count (where weight of the output value is number of trees which have output that value) method is employed to assign weightage to each output. Then the output with the highest weightage is recorded as the final output.

5.1. Forest of ANN

Applying the same concept of Bagging, the data sets were created and 11 ANNs were trained on these data sets. The learning rate and architecture of all networks was same. Weightage-by-count method was employed in voting scheme to predict exact outputs.

5.2. LTS Using AANN

Unlike above all methods, Auto Associative Neural Network is a novel non- discriminative ML technique in which outputs are same as inputs. Nonlinear Autoassociators perform pattern autoassociation of clustered regions. They have ability to capture the data distribution in multidimensional feature space. They are successfully used for speaker verification and speech recognition tasks [15]. We tested AANN for their performance on LTS rules.

Training AANNs: The training data was segregated in two data sets. In first one the inherent vowel 'Schwa' was supressed. In second one it was retained. All the features were used for this experiment. Two different neural networks with same input output dimensionality were trained on this data. The architecture of these networks was 120 L 120 N 120 N. Where L stands for linear activation function and N for nonlinear tangent sigmoidal activation function. They were trained for 100 iterations each with learning rate of 0.0001.

Testing AANNs: Each test vector was fed to each of these 2 neural networks. Then the euclidean distance between input and output was calculated. The network which gives smaller euclidean distance determines the class of that training vector.

6. Experimental Analysis & Results

We carried out the experimental comparison of various machine leraning techniques for LTS rules. Here we study the forest approaches on CART and ANNs and show how the forest approach is more robust as compared to normal ML approaches of decision tree and ANNs. Then we also show, the effect of adding syllabic features to original contextual features, on performance of forest based techniques.

Table 2: *Experiments on Hindi with CART, Decision Forest and Forest of AAN*

Context-Level	CART (Stop Value)			Decision-Forest	Forest of ANN
	1	5	10		
2L	89.59%	89.76%	89.20%	90.67%	92.14%
3L	90.47%	90.44%	90.63%	91.87%	92.37%
4L	91.09%	91.63%	91.54%	92.19%	92.92%

Table 3: *Subjective Evaluations*

Test	Without LTS	With LTS	Both Similar
AB-Test	7 / 40	27 / 40	6 / 40
MOS	3.4	3.8	-

6.1. Experiments With Contextual Features

We experimented with different levels of contextual features like 2, 3 and 4 (2L, 3L and 4L) using CART, ANN, Decision Forest and Forest of ANNs algorithms respectively, to know which of these levels contribute more to LTS. Results in the Table-2 show that 4-level contextual features gave the best performance.

6.2. Experiments with CART and ANN

Various stop values for CART were tried for optimal performance. Table 2 shows that stop value of 5 performed better. Best performance of CART was 91.63% The ANNs show slightly better performance than CART of 92.71%. Because of their highly connected nature, it seems that ANNs outperform CART.

6.3. Experiments with Decision Forest and Forest of ANNs

The results in Table 2 also indicate that Forest techniques always performs better than normal CART and ANN. The 4-level contextual feature was given more importance, as it gives better performance than 2 and 3-level contextual features. The performance of Decision Forest and ANN Forest was observed to be 92.19% and 92.92% respectively. Although reasons for the better performance of Decision Forest was explained in [14], to prove this statistically, we conducted another experiment by building separate models of CART and Decision Forest with partitioned training data like 10, 20, 30.. 100% of training data and evaluated these models with test data. The Fig 1 shows comparison between DF and CART algorithms. From the Fig 1, it is observed that increasing the training data leads to decline in CART performance because of over-fitting. While data overfitting is automatically nullified in Decision forest because of it's inherent multiplicity. So it always gives better performance. The same is true for Forest of ANNs.

6.4. Experiments with Syllabic Features

Indian Languages have generated from Brahmi scripts and these languages are syllabic in nature. So, syllabic features like Syllabic structure, Schwa Position, Onset of syllable and Number of Syllables in that word contribute to improve the performance. We have tested the performance of Schwa deletion by adding syllabic features to contextual feature set. Table 4 shows improvement in performance using contextual and syllabic features.

Ten Hindi sentences were synthesized with and without ap-

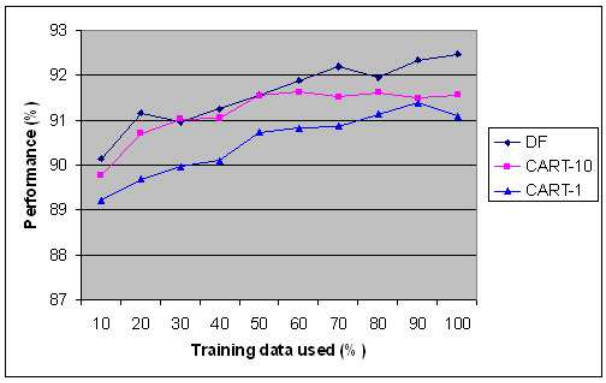

Figure 1: *Performance of CART (Stop values - 1 and 10) and Decision Forest (DF) with variable training data*

Table 4: *Performance of Hindi LTS with Contextual and Syllabic features*

Features Used	Phone-Accuracy	Word-Accuracy
Decision Forest with Contextual Features	92.19%	86.25%
Decision Forest with Contextual+Syllabic Features	92.86%	87.25%
ANN Forest with Contextual Features	92.92%	88.10%
ANN Forest with Contextual+Syllabic Features	93.18%	88.67%

plying LTS rules using Forest of AAN approach. Table 3 shows subjective evaluation using MOS (score between 1 (worst) to 5 (best)) and AB-Test (choice between two instances of same sentence) done by four subjects.

6.5. Experiments on AANN

Since contextual and syllabic feautures together proved to be the best in earlier experiments, we tested the same data on AANN. They could achieve performance of 86.37%. The failure of AANN to perform at par with other techniques points out the important fact that, LTS rules do not form limited number prominent clusters in input feature space. Because these rules are complex with many exceptions, they make input space very sparsely distributed with tiny clusters.

6.6. Comparison with Rule based system

State-of-the-art rule based algorithm for Hindi LTS is given in [2] by M. Choudhury. To compare our results with that algorithm, we used the same algorithm without morphological analyzer on our test data set, and observed the performance to be 88.17%. Table-5 shows the comparisons for Rule based system and Decision Forest System.

7. Conclusion

In this paper, we compared various ML techniques with various contextual features. First we experimented with various levels of contextual features like 2, 3 and 4 level using CART, ANN, Decision Forest and ANN Forest and observed that 4 level contextual features were better, compared to other levels used. We also evaluated performance of CART with different stop values.

Table 5: *Performance of Hindi LTS with Rule-based algorithm, CART, Decision Forest and Forest of ANN*

System	Phone Accuracy	Word Accuracy
Rule-based	88.17%	80.98%
CART (Stop-5)	91.63%	86.03%
Decision Forest (DF)	92.86%	87.25%
Forest of ANN	93.18%	88.67%

Secondly we studied performances of forest-based techniques and found that Decision Forest and ANN Forest always performs better than their non-forest counterparts. This fact reflects their immunity towards over-fitting and noisy data. We experimented with different syllabic features like syllabic structure, onset of syllable, Schwa position and number of syllables in that word and we achieved better performance with these syllabic structures. Finally we tested non discriminative AANN technique which could not perform well in comparison to discriminative ones. To conclude, the paper shows that forest of ANN approach performs best, among the selected approaches, with accuracy of 93.18% for Hindi LTS rules.

8. References

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

[2] Choudhury, M., "Rule-Based Grapheme to Phoneme Mapping for Hindi Speech Synthesis", 90th Indian Science Congress of the International Speech Communication Association (ISCA), Bangalore, 2003.

[3] Monojit Choudhury, Anupam Basu and Sudeshna Sarkar, "A Diachronic Approach for Schwa Deletion in Indo Aryan Languages", Proceedings of SIGPHON

[4] Anjan Banerjee, Monojit Choudhury, Sudeshna Sarkar and Anupam Basu, "Learning Schwa Pronounceability Rules in Bengali Compound Words using Decision Trees",

[5] http://speech.iiit.ac.in/ speech/speechdemo.html, "ITrans-3 notation".

[6] Brieman, L., "Classification and Regression Trees", Chapman & Hall/CRC, 1984.

[7] Thomas M. Mitchell, "Machine Learning", McGraw-Hill Higher Education, 1997.

[8] Yvon, F., "Self-learning techniques for grapheme-to-phoneme conversion", Proceeding of the 2nd Onomastica Research Colloquium, London, Nov, 1994.

[9] Black, A.W. and Lenzo, K. and Pagel, V., "Issues in Building General Letter to Sound Rules", International Speech Communication Association, 1998.

[10] Weijters, T. Thole, J., "Speech synthesis with artificial neural networks", Neural Networks, 1993

[11] Mana, F. and Massimino, P. and Pacchiotti, A., "Using Machine Learning Techniques for Grapheme to Phoneme Transcription",

[12] B. Yegnanarayana, "Artificial Neural Networks", New Delhi: Prentice Hall of India, 1999.

[13] Tong, W. and Hong, H. and Fang, H. and Xie, Q. and Perkins, R. "Decision forest: combining the predictions of multiple independent decision tree models", J. Chem. Inf. Comput. Sci, Volume 43, 2003

[14] Brieman, L., "Bagging predictors", Journal of Machine Learning, Volume 24, 1996.

[15] S. P. Kishore and B. Yegnanarayana "Speaker verification: Minimizing the channel effects using autoassociative neural network models", Proceedings of IEEE Int. Conf. Acoust., Speech, and Signal Processing, (Istanbul), pp. 11011104, 2000.