Text Processing for Text to Speech Systems in Indian Languages

by

Anand Arokia Raj, Tanuja Sarkar, Sathish Chandra Pammi, Santhosh Yuvaraj, Mohit Bansal, Kishore Prahallad, Alan W Black

in

in Proceedings of 6th ISCA Speech Synthesis Workshop SSW6, Bonn, Germany, 2007.

Report No: IIIT/TR/2007/32

Centre for Language Technologies Research Centre
International Institute of Information Technology
Hyderabad - 500 032, INDIA
July 2007

Text Processing for Text-to-Speech Systems in Indian Languages

Anand Arokia Raj [1], *Tanuja Sarkar* [1], *Satish Chandra Pammi* [1],
Santhosh Yuvaraj [1], *Mohit Bansal* [2], *Kishore Prahallad*[1 3], *Alan W Black*[3]

[1] International Institute of Information Technology, Hyderabad, India.
[2] Indian Institute of Technology, Kanpur, India.
[3]Language Technologies Institute, Carnegie Mellon University, Pittsburgh, USA.
skishore@cs.cmu.edu, awb@cs.cmu.edu

Abstract

To build a natural sounding speech synthesis system, it is essential that the text processing component produce an appropriate sequence of phonemic units corresponding to an arbitrary input text. In this paper we discuss our efforts in addressing the issues of Font-to-Akshara mapping, pronunciation rules for Aksharas, text normalization in the context of building text-to-speech systems in Indian languages.

1. Introduction

The objective of a text to speech system is to convert an arbitrary given text into a corresponding spoken waveform. Text processing and speech generation are two main components of a text to speech system. The objective of the text processing component is to process the given input text and produce appropriate sequence of phonemic units. These phonemic units are realized by the speech generation component either by synthesis from parameters or by selection of a unit from a large speech corpus. For natural sounding speech synthesis, it is essential that the text processing component produce an appropriate sequence of phonemic units corresponding to an arbitrary input text.

One of the question often asked by end-users is why we don't have TTS systems for all or many of the 23 official Indian languages. What are the complexities: Is it because the synthesis technology isn't matured enough to be able to build for any language or is it because of the non-existence of speech databases in Indian languages?. Unfortunately, for a decade the core speech generation technology i.e., generation of speech from a phonemic sequence has largely been automated due to unit selection techniques [1]. With the introduction of statistical parametric speech synthesis techniques, it is much easier to build a voice in a language with fewer sentences and a smaller speech corpus [2] [3].

It is difficult to convince an end-user that the input to a TTS system is not a phonemic sequence but rather the raw text as available in news websites, blogs, documents etc which contain the required text in font-encodings, native scripts and non-standard words such as addresses, numbers, currency etc. The majority of the issues are associated in building a TTS for a new language is associated with handling of real-world text [4]. Current state-of-art TTS system in English and other well-researched languages use such rich set of linguistic resources such as word-sense disambiguation, morphological analyzer, Part-of-Speech tagging, letter-to-sound rules, syllabification, stress-patterns in one form or the other to build a text processing component of a TTS system. However for minority languages (which are not well researched or do not have enough linguistic resources), it involves several complexities starting from accumulation of text corpora in digital and processable format. Linguistic components are not available in such rich fashion for all languages of the world. In practical world, minority languages including some of the Indian languages do not have that luxury of assuming some or any of the linguistic components.

The purpose of this paper is to describe our efforts at IIIT Hyderabad to build a generic framework for build text processing modules and linguistic resources which could be extended to all of the Indian languages with minimal efforts and time. Our approach is to make use of minimal language information (i.e., information available with an average educated native speakers), take the aid of acoustic data and machine learning techniques [5]. In this paper we summarize some of our efforts in this direction but mainly for font identification, Font-to-Akshara conversion, pronunciation rules for Aksharas and text normalization.

2. Nature of Indian Language Scripts

The scripts in Indian languages have originated from the ancient Brahmi script. The basic units of the writing system are referred to as *Aksharas*. The properties of Aksharas are as follows: (1) An Akshara is an orthographic representation of a speech sound in an Indian language; (2) Aksharas are syllabic in nature; (3) The typical forms of Akshara are V, CV, CCV and CCCV, thus have a generalized form of C*V.

The shape of an Akshara depends on its composition of consonants and the vowel, and sequence of the consonants. In defining the shape of an Akshara, one of the consonant symbols acts as pivotal symbol (referred to as semi-full form). Depending on the context, an Akshara can have a complex shape with other consonant and vowel symbols being placed on top, below, before, after or sometimes surrounding the pivotal symbol (referred to as half-form).

Thus to render an Akshara, a set of semi-full or half-forms have to be rendered, which in turn are rendered using a set of basic shapes referred to as *glyphs*. Often a semi-full form or half-form is rendered using two or more glyphs, thus there is no one-to-one correspondence between glyphs of a font and semi-full or half-forms [6].

2.1. Convergence and Divergence

There are 23 official languages of India, and all of them except English and Urdu share a common phonetic base, i.e., they share a common set of speech sounds. While all of these languages share a common phonetic base, some of the languages

such as Hindi, Marathi and Nepali also share a common script known as Devanagari. But languages such as Telugu, Kannada and Tamil have their own scripts.

The property that makes these languages separate can be attributed to the phonotactics in each of these languages rather than the scripts and speech sounds. phonotactics is the permissible combinations of phones that can co-occur in a language.

2.2. Digital Storage of Indian Language Scripts

There is a chaos as far as the text in Indian languages in electronic form is concerned. Neither can one exchange the notes in Indian languages as conveniently as in English language, nor can one perform search easily on texts in Indian languages available over the web. This is because the texts are being stored in ASCII font dependent glyph codes as opposed to Unicode.

The glyph coding schemes are typically different for different languages and within a language there could exists several font-types with their own glyph codes (as many as major news-portals in a language). To view the websites hosting the content in a particular font-type, these fonts have to be installed on local machine. As this was the technology existed before the era of Unicode and hence a lot of electronic data in Indian languages were made and available in that form [7].

2.3. Need for Handling Font-Data

The text available in a font-encoding (or font-type) is referred to as *font-data*. While Unicode based news-portals and webpages are increasing, there are two main reasons to deal with ASCII based font-data: 1) Given that there are 23 official Indian languages, and the amount of data available in ASCII based font-encodings is much larger than the text content available in Unicode format, 2) If a TTS system has to read the text from a ASCII font based website then the TTS system should automatically identify the font-type and process the font-data to generate speech.

2.4. A Phonetic Transliteration Scheme for Digital storage of Indian Language Scripts

To handle diversified storage formats of scripts of Indian languages such as ASCII based fonts, ISCII (Indian Standard code for Information Interchange) and Unicode etc, it is useful and becomes necessary to use a meta-storage format.

A transliteration scheme maps the Aksharas of Indian languages onto English alphabets and it could serve as meta-storage format for text-data. Since Aksharas in Indian languages are orthographic represent of speech sound, and they have a common phonetic base, it is suggested to have a phonetic transliteration scheme such as IT3 [8] [6]. Thus when the font-data is converted into IT3, it essentially turns the whole effort into font-to-Akshara conversion.

3. Identification of Font-Type

Given a document we often need to identify the font-type, and sometimes a document can contain the data encoded in different font-types. Then the task would boil down to identifying the font-type for each line or for each word. In this paper, we propose the use of TF-IDF approach for identification of font-type. The term frequency - inverse document frequency (TF-IDF) approach is used to weigh each glyph-sequence in the font-data according to how unique it is. In other words, the TF-IDF approach captures the relevancy among glyph-sequence and font-type. In this approach, the term refers to a 'glyph' and the document refers to the font-data of a particular 'font-type'. Here the glyph-sequence could mean a single glyph or 'current and next' glyph or 'previous, current and next' glyph etc.

To build a document for each font-type, a web-site for each font-type was manually identified and around 0.12 million unique words were crawled for each of the font-type. The set of unique words for each font-type are referred to as a document representing the particular font-type. Thus given N documents (each representing a font-type), we considered three different terms namely, a single glyph or *current and next* glyph or *previous, current and next* glyph. For each term a TF-IDF weight was obtained as follows: (i) Calculate the term frequency for the glyph-sequence: The number of times that glyph-sequence occurred divided by the total number of glyph-sequences in that specific document. (ii) Calculate document frequency: In how many different documents (font-types) that specific glyph-sequence has occurred. (iii) Calculate inverse document frequency of the term and take logarithm of inverse document frequency.

To identify the font-type of a given test font-data, the steps involved are as follows: 1) Generate the terms (glyph-sequences) of the test font-data 2) Compute the relevancy scores of the terms and for each of the document (font-type) using the corresponding TF-IDF weights of the terms 3) The test font-data belongs to the document (font-type) which produces a maximum relevancy score.

The performance of TF-IDF approach for identification of font-type was evaluated on 1000 unique sentences and words per font-type. We have added English data as also one of the testing set, and is referred to as English-text. The performance of font-type identification system using different terms *single* glyph, *current and next* glyphs, *previous, current and next* glyphs are shown in Table 1, Table 2 and Table 3 respectively and it could be observed that the use of *previous, current and next* glyphs as a term provided an accuracy of 100% in identification of font-type even at the word level.

Table 1: *Performance of Single glyph based font models*

Font Name	Sentence-Level	Word-Level
Amarujala (Hindi)	100%	100%
Jagran (Hindi)	100%	100%
Webdunia (Hindi)	100%	0.1%
SHREE-TEL (Telugu)	100%	7.3%
Eenadu (Telugu)	0%	0.2%
Vaarttha (Telugu)	100%	29.1%
Elango_Panchali (Tamil)	100%	93%
Amudham (Tamil)	100%	100%
SHREE-TAM (Tamil)	100%	3.7%
English-text	0%	0%

4. Font-to-Akshara Mapping

Font-data conversion can be defined as converting the font encoded data into Aksharas represented using phonetic transliteration scheme such as IT3. As we already mentioned that Aksharas are split into glyphs of a font, and hence a conversion from font-data has essentially to deal with glyphs and model how a sequence of glyphs are merged to form an Akshara. As there exist many fonts in Indian languages, we have designed a generic framework has been designed for the conversion of font-data. It

Table 2: *Performance of current and next glyph based font models*

Font Name	Sentence-Level	Word-Level
Amarujala (Hindi)	100%	100%
Jagran (Hindi)	100%	100%
Webdunia (Hindi)	100%	100%
SHREE-TEL (Telugu)	100%	100%
Eenadu (Telugu)	100%	100%
Vaarttha (Telugu)	100%	100%
Elango_Panchali (Tamil)	100%	100%
Amudham (Tamil)	100%	100%
SHREE-TAM (Tamil)	100%	100%
English-text	100%	96.3%

Table 3: *Performance of previous, current and next based font models*

Font Name	Sentence-Level	Word-Level
Amarujala (Hindi)	100%	100%
Jagran (Hindi)	100%	100%
Webdunia (Hindi)	100%	100%
SHREE-TEL (Telugu)	100%	100%
Eenadu (Telugu)	100%	100%
Vaarttha (Telugu)	100%	100%
Elango_Panchali (Tamil)	100%	100%
Amudham (Tamil)	100%	100%
SHREE-TAM (Tamil)	100%	100%
English-text	100%	100%

has two phases, in the first phase we are building the base-map table for a given font-type and in the second phase forming and ordering the assimilation rules for a specific language.

4.1. Building a Base-Map Table for a Font-type

The base-map table provides the mapping basic between the glyphs of the font-type to the Aksharas represented in IT3 transliteration scheme. The novelty in our mapping was that the shape of a glyph was also included in building this mapping table. The shape of a glyph is dictated by whether it is rendered as pivotal consonant, or on top, bottom, left or right of the pivotal consonant. Thus the pivotal glyphs were appended with 0 (for full characters such as e, ka) or 1 (for half consonants such as k1, p1), '2' for glyphs occur at left hand side of a basic character (ex: i2, r2), '3' for glyphs occur at right hand side of a basic character (ex: au3, y3), '4' for glyphs occur at top of a basic character (ex: ai4, r4) and '5' for glyphs occur at bottom of a basic character (ex: u5, t5).

4.2. Forming Assimilation Rules

In the conversion process the above explained basic-mapping table will be used as the seed. A well defined and ordered set of assimilation rules have to be formed for each and every language. Assimilation is the process of merging two or more glyphs and generating a valid single character. This assimilation happens at different levels and our observation across many languages was that the firing of following assimilation rules were universally applicable. The rules are:(i) Modifier Modification, (ii) Language Preprocessing, (iii) Consonant Assimilation, (iv) Maatra Assimilation, (v) Consonant-Vowel Assimilation, (vi) Vowel-Maatra Assimilation, (vii) Consonants Clustering and (viii) Schwa Deletion.

The Modifier Modification is the process where the characters get modified because of the language modifiers like virama and nukta (ka + virama = k1). The Language Preprocessing step deals with some language specific processing like (aa3 + i3 = ri in Tamil) and (r4 moves in front of the previous first full consonant in Hindi). The Consonant Assimilation is known as getting merged two or more consonant glyphs and forms a valid single consonant like (d1 + h5 = dh1 in Telugu). The Maatra Assimilation is known as getting merged two or more maatra glyphs and forms a valid single maatra like (aa3 + e4 = o3 in Hindi). The Consonant-Vowel Assimilation is known as getting merged two or more consonant and vowel glyphs and forms a valid single consonant like (e + a4 + u5 = pu in Telugu). The Vowel-Maatra Assimilation is known as getting merged two or more vowel and maatra glyphs and forms a valid single vowel like (a + aa3 = aa in Hindi). The Consonant Clustering in known as merging the half consonant which usually occurs at the bottom of a full consonant to that full consonant like (la + l5 = lla in Hindi). The Schwa Deletion is deleting the inherent vowel 'a' from a full consonant in necessary places like (ka + ii3 = kii).

4.3. Testing and Evaluation

The evaluation on these font converters is carried out in two phases. We picked up three different font-types for training or forming the assimilation rules and one new font-type for testing per language. In the first phase for the selected three font-types the assimilation rules are formed and refined. In the second phase we chose a new font-type and built the base-map table only and used the existing converter without any modifications. We have taken 500 unique words per font-type and generated the conversion output. The evaluation results in Table 4 show that the font converter performs consistently even for a new font-type. So it is only sufficient to provide the base-map table for a new font-type to get a good conversion results. The issue of Font-to-Akshara mapping has been attempted in [7] and [9] but we believe that our framework is a generic one which could easily be extended to a new font-type with $> 99\%$ conversion accuracy.

5. Building Pronunciation Models For Aksharas

Having converted the font-data into Aksharas, the next step is to obtain appropriate pronunciation for each of the Aksharas. As noted earlier, Aksharas are orthographic representation of speech sounds and it is commonly believed or quoted that there is direct correspondence between what is written and what is spoken in Indian languages, however, there is no one-to-one correspondence between what is written and what is spoken. Often some of the sounds are deleted such as Schwa deletion in Hindi. Schwa is the default short vowel /a/which is associated with a consonant, and often it is deleted to aid in faster pronunciation of a word. Similarly there exists exceptions for Bengali and Tamil. There are attempts to model these exceptions in the form of the rules, however, they are often met with limited success or they use linguistic resources such as Morph analyzer. Such linguistic resources may not always be available for minority languages. Thus we had built a framework based on machine learning techniques where pronunciation of Aksharas could be modeled using machine learning techniques and using a small set of supervised training data.

Table 4: *Performance results for font conversion in Indian languages*

Language	Font Name	Training/Testing	Accuracy
Hindi	Amarujala	Training	99.2%
	Jagran	Training	99.4%
	Naidunia	Training	98.8%
	Webdunia	Training	99.4%
	Chanakya	Testing	99.8%
Marathi	Shree Pudhari	Training	100%
	Shree Dev	Training	99.8%
	TTYogesh	Training	99.6%
	Shusha	Testing	99.6%
Telugu	Eenadu	Training	93%
	Vaartha	Training	92%
	Hemalatha	Training	93%
	TeluguFont	Testing	94%
Tamil	Elango Valluvan	Training	100%
	Shree Tam	Training	99.6%
	Elango Panchali	Training	99.8%
	Tboomis	Testing	100%
Kannada	Shree Kan	Training	99.8%
	TTNandi	Training	99.4%
	BRH Kannada	Training	99.6%
	BRH Vijay	Testing	99.6%
Malayalam	Revathi	Training	100%
	Karthika	Training	99.4%
	Thoolika	Training	99.8%
	ShreeMal	Testing	99.6%
Gujarati	Krishna	Training	99.6%
	Krishnaweb	Training	99.4%
	Gopika	Training	99.2%
	Divya	Testing	99.4%

5.1. Creation of Data-set

Given the input word list with the corresponding pronunciations in terms of phones, feature vectors were extracted for training the pronunciation model at the phone level. About 12200 sentences in IT3 format were used to collect the training data, for building the pronunciation model in Hindi. These sentences had about 26000 unique words, which were used to extract around 32800 feature vectors. Different sets of feature vectors to experiment on the selection of features. As for Bengali and Tamil, 5000 words with corresponding pronunciations were used for obtaining about 9000 feature vectors.

5.2. Use of Contextual Features

Contextual features refers to the neighbor phones in a definite window-size/level. Using the contextual features, experiments were performed for various Contextual Levels (CL). A decision forest was built for each phone to model its pronunciation. A decision forest is a set of decision trees built using overlapping but different sub-sets of the training data and it employs a majority voting scheme on individual prediction of different trees to predict the pronunciation of a phone. Table 5 shows the results of pronunciation model for Hindi, Bengali and Tamil using various level of contextual features. We found that that a context level of 4 (i.e., 4 phones to the left and 4 phones to the right) was sufficient to model the pronunciation and moving beyond the level of 4, the performance was degraded.

Table 5: *Pronunciation Model with Contextual features*

Languages	Context Level			
	2	3	4	6
Hindi	90.24%	91.44%	**91.78%**	91.61%
Bengali	82.77%	84.48%	**84.56%**	83.56%
Tamil	98.16%	**98.24%**	98.10%	98.05%

5.3. Acoustic-Phonetic and Syllabic Features

Acoustic phonetic features lists the articulatory properties of the consonants and the vowels. Typically vowels are characterized by the front, back, mid position of the tongue while consonants are characterized by manner and place of articulation and voicing and nasalization features. Syllabic features indicate where a particular syllable is of type CV or CCV, or CVC etc. The performance of the pronunciation model for Hindi, Tamil and Bengali using syllabic and acoustic-phonetic features of the current and neighboring phones are shown in Table 6 and Table 7 respectively. We found that the use of syllabic or acoustic-phonetic features didn't show any significant improvement than that of contextual features for Hindi, Tamil and Bengali.

A rule based algorithm for Hindi LTS is given in [10]. To compare our results with the rule-based algorithm, we have used the same algorithm with out morphological analyzer on our test data set. We found that the performance of pronunciation model using rule-based technique was 88.17%. while the decision forest model in Table 6 was providing an accuracy of 92.29%.

Table 6: *Pronunciation Model with Syllabic features*

Feature Sets	Languages		
	Hindi	Bengali	Tamil
Syl_Struct. of Cur. Phone	**92.29%**	**82.41%**	**98.31%**
Syl_Struct. of all Phones	91.61%	67.56%	98.27%

Table 7: *Pronunciation Model with Acoustic-Phonetic features*

Feature Sets	Languages		
	Hindi	Bengali	Tamil
Acoustic_Phonetic	89.73%	**84.78%**	**98.18%**
+ Syl_Struct. of Curr. Phone	89.73%	81.21%	98.17%
+ Syl_Struct. of all Phones	**91.09%**	69.33%	**98.13%**

6. Normalizing of Non-Standard Words

Unrestricted texts include Standard Words (common words and Proper Names) and Non-Standard Words (NSWs). Standard Words have a specific pronunciation that can be phonetically described either in a lexicon, using a disambiguation processing to some extent, or by letter-to-sound rules. In the context of TTS the problem is to decide how an automatic system should pronounce a token; even before the pronunciation of a token, it

Table 8: *Taxonomy of NSWs with examples*

Category	Description	Examples
Addr	Address (house/street no.)	12/451 Janapath Road
Curr	Currency	Rs. 7635.42
Count	Count of items	10 computers, 500 people
Date	Date(to be expanded)	1/1/05, 1997-99
PhoneNo	As sequence of digits	040 2300675
Pin	As sequence of digits	208023
Score	Cricket, tennis scores	India 123/4, sets 3-5 3-4 5-6
Time	Time (to be expanded)	1.30, 10:45-12:30, 11.12.05, 1930 hrs
Units	As decimal or number	10.5 kms, 98 %, 13.67 acres
NUM	Default category	

is important to identify the NSW-Category of a token. A typical set of NSW-category and their examples are shown in Table 8.

6.1. Creation of Supervised Training Data

To build a training dataset, it typically requires a large manual effort to annotate an example with the appropriate NSW-category. For example, given a word corpus $> 3M$ words in Telugu, Tamil and Hindi, we extracted 150-500K sentences containing an NSW. Annotating such huge set of examples needs lots of time and effort. To minimize such effort, we used a novel frequency based approach to create a representative example set.

NSW techniques uses context information for disambiguation with various window sizes, context information contains a set of word like units which occurs in left and right side of a NSW, and this information is to be considered as a features characterizing a NSW. However, not of all context would be useful, so we used a window size of 2 (left and right) as a default and given to the pattern generator module. The pattern generator takes the four tokens (two to left and two to the right of a NSW) and generates 15 patterns using all possible combinations of 4(like examples, 0001, 0010, 0011, 0100, ., 111) where 1 represent presence of a token and 0 represent deletion of the token. Given such 15 patterns for each example, these patterns were sorted in the descending order of their frequency and based on a threshold a set of patterns were choosen and given to a native speaker to annotate the NSW category. The user interface was built such that if the native speaker couldn't annotate the NSW with the given pattern, then an extended context was presented to him at varying levels. Using the frequency based approach, we could reduce the training examples to around 1000-1500 which a native could annotate within a couple of hours. Having got the annotation done, we looked at level of context the native speaker has used to annotate a NSW. We found less than 10% of time the user has looked into a context information more than a window size of two.

6.2. Performance of Base-line System

Using word level units and decision tree, we built a base-line system to predict the category of a NSW. We have tested the performance of the system on a separate manually prepared data obtained from a different source (web) referred to as Test-Set-1 (TS1). The results of prediction of NSW category on TS1 is shown in Table 9.

Table 9: *Performance of prediction of NSW-category Using Word Level Features*

Language	% accuracy on Training set	% accuracy on TS1
Telugu	99.57%	63.52%
Hindi	99.80%	66.99%
Tamil	99.01%	55.42%

Table 10: *Performance of prediction of NSW-category Using Syllable level Features*

Language	% accuracy on Training set	% accuracy on TS1	Diff with base-line
Telugu	99.57%	91.00%	27.48%
Hindi	99.80%	82.80%	15.81%
Tamil	99.01%	87.20%	31.78%

The performance of the base-line system on TS1 is around 60%. After analyzing the errors made by the system, we found that the errors are primarily due to new words found in the context of NSW, and Indian languages being rich in inflectional and derivative morphology, the roots of many of these words were present in the training data. It suggests that we should use roots of the context as the features to predict NSW-category, however, such approach needs morphological analyzers. Many of the Indian languages fall into category of minority languages where linguistic resources are scarce. Thus we wanted to investigate sub-word units such as syllables and their combinations as features for prediction of NSW-category.

Our experiments on POS-tagging on Hindi, Bengali and Telugu using syllable-level units further provided evidence that syllable level features could be used as alternative and a first-order approximation of root of a word [11]. After initial set of experiments to explore different possibilities of using syllable-level features, we confined to a set of following three syllable level features. They are: 1) F1: previous ten and next ten syllables of a NSW, 2) F2: previous ten and next ten syllables and onset of each syllables and 3) F3: Onset, vowel and coda of previous ten and next ten syllables.

Using decision forest, the final prediction of NSW-category is chosen based on voting on the outputs of the three decision trees built using F1, F2 and F3. This strategy gets the results of each decision tree and performs a majority voting to predict the NSW-category. The performance of the decision forest based system using syllable level features is shown in Table 10. We found that the results of using syllable-level features for text normalization performed significantly better than that of using word-level features. This significant improvement in the performance is primarily due to syllables acting a first-order approximation of roots of the context words and thus minimizing the problem of unseen context. The final performance of the text normalization system is further improved after using expander module from 91.00%, 82.80% and 87.20% to 96.60%, 96.65% and 93.38% for languages Telugu, Hindi and Tamil respectively.

7. Conclusions

This paper explained the nature and difficulties associated with building text processing components of TTS systems in Indian languages. We have discussed the relevancy of font-

identification and font-to-Akshara conversion and proposed a TF-IDF based approach for font-identification. A novel approach of conversion from font-to-Akshara using the shapes of the glyphs and the assimilation rules was explained. We have also studied the performance of pronunciation models for different features including contextual, syllabic and acoustic-phonetic features. Finally we have shown that syllable-level features could be used to build a text normalization system whose performance is significantly better than the word-level features.

8. References

[1] Hunt A.J. and Black A.W., "Unit selection in a concatenative speech synthesis system for a large speech database," in *Proceedings of IEEE Int. Conf. Acoust., Speech, and Signal Processing*, 1996, pp. 373–376.

[2] Black A.W., Zen H., and Tokuda K., "Statistical parametric speech synthesis," in *Proceedings of IEEE Int. Conf. Acoust., Speech, and Signal Processing*, Honolulu, USA, 2007.

[3] Zen H., Nose T., Yamagishi J., Sako S., Masuko T., Black A.W., and Tokuda K., "The hmm-based speech synthesis system version 2.0," in *Proc. of ISCA SSW6*, Bonn, Germany, 2007.

[4] Sproat R., Black A.W., Chen S., Kumar S., Ostendorf M., and Richards C., "Normalization of non-standard words," *Computer Speech and Language*, pp. 287–333, 2001.

[5] HaileMariam S. and Prahallad K., "Extraction of linguistic information with the aid of acoustic data to build speech systems," in *Proceedings of IEEE Int. Conf. Acoust., Speech, and Signal Processing*, Honolulu, USA, 2007.

[6] Prahallad L., Prahallad K., and Ganapathiraju M., "A simple approach for building transliteration editors for indian languages," *Journal of Zhejiang University Science*, vol. 6A, no. 11, pp. 1354–1361, 2005.

[7] Garg H., *Overcoming the Font and Script Barriers Among Indian Languages*, MS dissertation, International Institute of Information Technology, Hyderabad, India, 2004.

[8] Ganapathiraju M., Balakrishnan M., Balakrishnan N., and Reddy R., "Om: One tool for many (Indian) languages," *Journal of Zhejiang University Science*, vol. 6A, no. 11, pp. 1348–1353, 2005.

[9] Khudanpur S. and Schafer C., "http://www.cs.jhu.edu / cschafer/ jhu_devanagari_cvt_ver2.tar.gz," 2003.

[10] Choudhury M., "Rule-based grapheme to phoneme mapping for hindi speech synthesis," in *90th Indian Science Congress of the International Speech Communication Association (ISCA)*, Bangalore, India, 2003.

[11] S. Chandra Pammi and Prahallad K., "POS tagging and chunking using decision forests," in *Proceedings of Workshop on Shallow Parsing in South Asian Languages, IJCAI*, Hyderabad, India, 2007.