THE UNIVERSITY of EDINBURGH

Edinburgh Research Explorer

DNN-based Speech Synthesis for Indian Languages from ASCII text

Citation for published version:
Ronanki, S, Gangireddy, SR, Bollepalli, B & King, S 2016, DNN-based Speech Synthesis for Indian Languages from ASCII text. in *9th ISCA Speech Synthesis Workshop.* pp. 74-79, 9th ISCA Speech Synthesis Workshop , Sunnyvale, California, United States, 13/09/16. https://doi.org/10.21437/SSW.2016-12

Digital Object Identifier (DOI):
10.21437/SSW.2016-12

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
9th ISCA Speech Synthesis Workshop

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 03. May. 2024

DNN-based Speech Synthesis for Indian Languages from ASCII text

Srikanth Ronanki[1], *Siva Reddy*[2], *Bajibabu Bollepalli*[3], *Simon King*[1]

[1]The Centre for Speech Technology Research, University of Edinburgh, United Kingdom
[2]ILCC, School of Informatics, University of Edinburgh, United Kingdom
[3]Department of Signal Processing and Acoustics, Aalto University, Finland

srikanth.ronanki@ed.ac.uk

Abstract

Text-to-Speech synthesis in Indian languages has a seen lot of progress over the decade partly due to the annual Blizzard challenges. These systems assume the text to be written in Devanagari or Dravidian scripts which are nearly phonemic orthography scripts. However, the most common form of computer interaction among Indians is ASCII written transliterated text. Such text is generally noisy with many variations in spelling for the same word. In this paper we evaluate three approaches to synthesize speech from such noisy ASCII text: a naive Uni-Grapheme approach, a Multi-Grapheme approach, and a supervised Grapheme-to-Phoneme (G2P) approach. These methods first convert the ASCII text to a phonetic script, and then learn a Deep Neural Network to synthesize speech from that. We train and test our models on Blizzard Challenge datasets that were transliterated to ASCII using crowdsourcing. Our experiments on Hindi, Tamil and Telugu demonstrate that our models generate speech of competetive quality from ASCII text compared to the speech synthesized from the native scripts. All the accompanying transliterated datasets are released for public access.

Index Terms: Indian Languages, Speech Synthesis, Deep Neural Networks, ASCII transliteration.

1. Introduction

Though a large number of Indian languages have indigenous scripts, the lack of a standardized keyboard, and the ubiquity of QWERTY keyboards, means that people most often write using ASCII[1] [1] text using spellings motivated largely by pronunciation [2]. Increasingly, many technologies such as Web search and natural language processing are adapting to this phenomenon [3, 4, 5]. In the area of Speech Synthesis, although the efforts of the 2013, 2014 and 2015 Blizzard Challenges[2] [6, 7] resulted in improvements to the naturalness of speech synthesis of Indian languages, the text was assumed to be written in native script. In this work, we transliterate Blizzard data to informal chat-style ASCII text using Mechanical Turkers, and synthesize speech from the resulting transliterated ASCII text. This represents a more realistic use case than in the Blizzard Challenge.

Synthesizing speech from ASCII text is challenging: Since there is no standard way to spell pronunciations, people often spell same word in multiple ways, e.g., the word *start* in Telugu can be ASCII spelled *prarhambham*, *prarambham*, *prarambam*, *praranbam*, etc. whilst words that differ in both pronunciation and meaning might be spelled the same, e.g., the words *ledhu* and *ledu* in Telugu could both be spelled *ledu*.

We address these problems by first converting ASCII graphemes to phonemes, followed by a DNN to synthesise the speech. We propose three methods for converting graphemes to phonemes. The first model is a naive model which assumes that every grapheme corresponds to a phoneme. In the second model, we enhance the naive model by treating frequently co-occurring character bi-grams as additional phonemes. In the final model, we learn a Grapheme-to-Phoneme transducer from parallel ASCII text and gold-standard phonetic transcriptions. The contributions of this paper are:

- to synthesize speech from ASCII transliterated text for Indian languages, which to our knowledge is the first such attempt. Our results show that our Grapheme-to-Phoneme conversion model combined with a DNN acoustic model performs competitively with state-of-the-art speech synthesizers that use native script text.
- the release of parallel ASCII transliterations of Blizzard data to foster research in this area.

2. Related work

2.1. Transliteration of Indian Languages

Many standard transliteration systems exist for Indian languages. Table 1 shows different transliterations for an example sentence. Among these, CPS (Common Phone Set) [8] and IT3 [9] are widely used by the speech technology community, ITRANS[3] [10] is used in publishing houses, and WX[4] [11] by the Natural Language Processing (NLP) community. Though these scripts provide umambiguous conversion to native Indian scripts, due to their lack of readability, and the overhead in learning how to use them, people still spell their words motivated by pronunciation. One such transliteration is shown in the row *Informal* of Table 1.

The most common trend observed in the literature is to treat transliteration as a machine translation and discriminative ranking problem [12]. Our work aims to exploit the fact that transliterations are phonetically motivated, and therefore treat transliteration as a conversion problem. Specifically, we convert informal transliterations to phonetic script, and then synthesize speech from the phonetic script using a DNN.

2.2. Statistical Speech Synthesis

Most existing work in speech synthesis for Indian languages uses unit selection [13] with syllable-like units [14, 15]. Re-

[1]The ASCII character set is the union of Roman alphabets, digits, and a few punctuation marks.

[2]http://www.synsig.org/index.php/Blizzard_Challenge

[3]https://en.wikipedia.org/wiki/ITRANS

[4]https://en.wikipedia.org/wiki/WX_notation

Table 1: Transliteration of Hindi text in various notations

Original Sentence	आपके हिंदी पसंद करने पर खुशी हुई
Notation	**Transliteration**
CPS	aapakei hiqdii pasaqda karanei para khushii huii
IT3	aapakei hin:dii pasan:da karanei para khushii huii
ITRANS	Apake hiMdI pasaMda karane para khushI huI
WX	Apake hiMdI pasaMda karane para KuSI huI
Informal	apke hindi pasand karne par kushi hui

Table 2: Training data for the Grapheme-to-Phoneme model

Word	Informal transliteration	Pronunciation (CPS notation)
కాంగ్రెస్	congress	/k/aa/q/g/r/e/s/
பிரவேசிக்கவும்	pravesikkavum	/p/i/r/a/w/ei/c/i/k/k/a/w/u/m/
आपके	aapke	/aa/p/a/k/e/

cently, based on the observation that Indian languages share many commonalities in phonetics, a language independent phone set was proposed, and was used in building statistical parametric (HMM-based) speech synthesis systems [8]. We make use of this common phone set in one of our models.

Our work also aligns with the recent literature on unsupervised learning for text-to-speech synthesis which aims to reduce the reliance on human knowledge and the manual effort required for building language-specific resources [16, 17, 18]. These approaches are able to learn from noisy input representations where there is no standard orthography. Following the success of DNNs for speech recognition [19] and synthesis [20, 21, 22], we also use a DNN as the acoustic model.

3. Our Approach

Our speech synthesis pipeline consists of two steps: 1) Converting the input ASCII transliterated text to a phonetic script; 2) learning a DNN based speech synthesizer from the parallel phonetic text and audio signal.

3.1. Converting ASCII text to Phonetic Script

We explore three different approaches which vary in the degree of supervision in defining a phoneme.

3.1.1. Uni-Grapheme Model

In this approach, we assume each ASCII grapheme acts as a phoneme. We assume that the DNN will learn to map these "phonemes" to speech sounds. We normalize the data to lowercase and remove all punctuation marks. This ensures that the phone-set contains 26 letters and an extra $/sil/$ phone to mark beginning and end of the sentence.

3.1.2. Multi-Grapheme Model

In this approach, in addition to uni-graphemes, we also include some frequently co-occurring bi-graphemes as "phonemes".

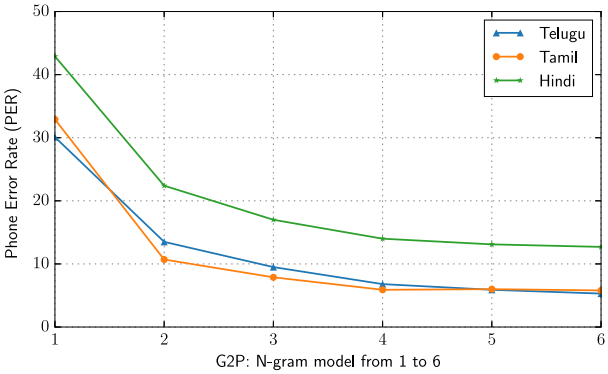

Figure 1: Performance of G2P models from uni-gram to 6-gram for Hindi, Telugu and Tamil

From manual inspection of the top 50 bi-graphemes, we found that the phonemes indicating stop consonants such as $/kh/$, $/ch/$, $/th/$, $/ph/$, $/bh/$ and long vowels such as $/aa/$, $/ii/$, $/ee/$, $/oo/$, $/uu/$ and dipthongs such as $/ai/$, $/au/$, $/ou/$ appear most frequently across languages. We selected 17 of these bi-graphemes as phonemes in addition to the above 27 uni-graphemes, making a total of 44 phonemes.

3.1.3. Grapheme-to-Phoneme (G2P) Model

In this model, we assume the phoneme-set is given. We use the common phone set (CPS, [8]) to work with the languages of interest. We convert the native text to CPS phonetic text using deterministic converters [9, 23]. We then align the phonetic transcriptions to the ASCII transliterations from Mechanical Turkers to create a pronunciation table. Table 2 shows the parallel data with the native text in the first column, the informal ASCII transliteration in the second column, and the CPS phonetic transcription in the third column.

Given the pronunciation lexicon, we train a G2P transducer [24] for each language separately with varying n-gram sequences. The corpus used for training is described in Section 4.1. Figure 1 displays the phone error rate of the G2P model with varying n-grams. The 6-gram model achieved the lowest phone error rate across the three languages. Telugu and Tamil achieved lower phonetic error rates compared to Hindi. This can be attributed to the ineffective handling of intricate *schwa deletion*, a well-known phenomenon in Indo-Aryan languages.

An advantage with this model is that, since the phoneme-set is standard, we can train G2P and DNN on two independent datasets – G2P on parallel transliterations of a very large corpus that could be obtained via crowdsourcing, and DNN model on gold phonetic speech transcriptions independently of the G2P model's performance. We leave this aspect of our work for future. In this work, we train a DNN model on the output of G2P aligned with natural speech.

3.2. DNN Speech Synthesizer

We use a DNN for learning to synthesize speech from the phonetic strings obtained in the previous step. We use two independent DNNs – one for duration and the other for acoustic modeling.

Let $x_i = [x_i(1), ..., x_i(d_x)]^T$ and $y_i = [y_i(1), ..., y_i(d_y)]^T$ be static input and output feature vectors of the DNN, where d_x and d_y denote the dimensions of x_i and y_i, respectively, and T denotes transposition.

Duration Model: For duration modeling, the input comprises binary features (x_p) derived from a subset of the questions used by the decision-tree clustering in the standard HTS synthesiser. Similar to [20, 21], frame-aligned data for DNN training is created by forced alignment using the HMM system. The output is an eight-dimensional vector (y_p) of durations for every phone, comprising five sub-state durations, the overall phone duration, syllable duration and whole word duration. We use this form of multi-task learning to improve the model; the three additional features (phone, syllable, and word durations) act as a secondary task to help the network learn more about suprasegmental variations in duration at word level. At synthesis time, these features are predicted, but ignored.

Acoustic Model: The input uses the same features as duration prediction, to which 9 numerical features are appended. These capture frame position in the HMM state and phoneme, state position in phoneme, and state and phoneme duration. The DNN outputs comprise MCCs, BAPs and continuous $logf_0$ (all with deltas and delta-deltas) plus a voiced/unvoiced binary value.

In both acoustic and duration model, all the input features are normalized to the range of $[0.01, 0.99]$ and output features are normalized to zero mean and unit variance. The DNNs are then trained to map the linguistic features of input text to duration and acoustic features respectively. If $D(x_i)$ denotes the DNN mapping of x_i, then the error of the mapping is given by:

$$\epsilon = \sum ||y_i - D(x_i)||^2 \tag{1}$$

$$D(x_i) = \widetilde{d}(z_{n+1}) \tag{2}$$

$$z_{n+1} = d(w^{(n)} d(z_n)) \tag{3}$$

$$d(\vartheta) = a \tanh(b\vartheta), \widetilde{d}(\vartheta) = \vartheta \tag{4}$$

where n indexes layer and $w^{(n)}$ is the weight matrix of the n^{th} layer of the DNN model.

At synthesis time, duration is predicted first, and is used as an input to the acoustic model to predict the speech parameters. Maximum likelihood parameter generation (MLPG) using pre-computed variances from the training data is applied to the output features for synthesis, and spectral enhancement post-filtering is applied to the resulting MCC trajectories. Finally, the STRAIGHT vocoder [25] is used to synthesize the waveform.

4. Experimental Setup

4.1. Speech Databases

Our languages of interest are Hindi, Tamil and Telugu, all of which are widely-spoken Indian languages. We train and test on the 2015 Blizzard Challenge data which contains about four hours of speech and corresponding text for each language. The data-set contains 1710 utterances for Hindi, 1462 utterances for Tamil, and 2481 utterances for Telugu, with a single speaker per language. We used 92% of the data for training, 4% for development and 4% for testing.

4.2. Annotation

Starting from the original transcriptions in native script, we asked crowdsourced human annotators to ASCII transliterate them using pronunciation as their main motivation for spelling. For Hindi and Tamil, we recruited paid workers via Mechanical Turk who could read and speak the language fluently (as self-reported); for Telugu we had access to a trusted pool of native speakers. We tokenize each sentence to words with whitespace and punctuations as the delimiters. The annotators were provided with a web-interface containing a text box for each word. This ensures transliteration of every word given in the input sentence. The total number of annotators for Telugu, Tamil and Hindi are 50, 66 and 82 respectively. We diversified train, dev and test splits by having different set of annotators for each split.

4.3. Experimental Settings

We used the same DNN architectures (Section 3.2) for both duration and acoustic modeling. The number of hidden layers used was 6 with each layer consisting of 1024 nodes. As shown in equation 4, the tanh function was used as the hidden activation function, and a linear activation function was employed at the output layer. During training, L2 regularization was applied to the weights with penalty factor of 0.00001, the mini-batch size was 256 for the acoustic model and 64 for the duration model. For the first 10 epochs, momentum was 0.3 with a fixed learning rate of 0.002. After 10 epochs, the momentum was increased to 0.9 and from that point on, the learning rate was halved at each epoch. The learning rate of the top two layers was always half that of other layers. Learning rate was fine-tuned in duration models to achieve best performance. The maximum number of epochs was set to 30 (i.e., early stopping).

4.4. Our Models

As outlined in Section 3.1, we train three different models for each language. The number of questions used in DNN were different from system to system. For Uni-Grapheme model (labelled as **UGM**), the questions based on quin-phone identity were used, and other questions include suprasegmental features such as syllable, word, phrase and positional features. For Multi-Grapheme model (labelled as **MGM**) and Grapheme-to-Phoneme model (labelled as **G2P**), other questions based on position and manner of articulation were additionaly included.

4.5. Benchmark

As a benchmark, we use the DNN speech synthesizer trained on CPS phonetic transcriptions of the speech data. The goal is thus to synthesize speech from ASCII text that is as close as possible in quality to this benchmark (labelled as **BMK**).

5. Results

5.1. Objective Evaluations

5.1.1. Duration Model

To evaluate the duration prediction DNN, we calculated the root-mean-square error (RMSE) and Pearson correlation between reference and predicted durations, where the reference durations are estimated from the forced-alignment step in HTS. Tables 4 and 5 present the results on test data.

Overall, the benchmark system showed better performance than other systems in all languages. Among the proposed approaches, G2P performed slightly better than the other two in terms of correlation, whereas RMSE performance was not consistent across the languages. A possible explanation for this is that G2P uses superior phone set defined manually whereas UGM and MGM use unsupervised phones. Nevertheless, the proposed systems are not too far from the benchmark.

Table 3: Objective results of the proposed techniques versus the benchmark approach. MCD and BAP are Mel-Cepstral Distortion and Band Aperiodicity distortion, respectively. V/UV error means frame-level voiced/unvoiced prediction error. Root Mean Squared Error (RMSE) of F0 was calculated on a linear frequency scale.

	MCD (dB)	**BAP (dB)**	**F0 RMSE (Hz)**	**V/UV error rate(%)**	**MCD (dB)**	**BAP (dB)**	**F0 RMSE (Hz)**	**V/UV error rate(%)**
Method	**Telugu**				**Hindi**			
Uni-Grapheme-TTS	4.97	2.05	35.13	6.23	4.82	1.92	10.87	9.25
Multi-Grapheme-TTS	5.02	2.06	36.49	6.01	4.81	1.92	11.29	9.31
G2P-TTS	4.77	2.04	35.18	5.94	4.80	1.92	10.77	9.30
Benchmark	4.81	2.04	37.09	5.83	4.52	1.90	10.86	8.07
Method	**Tamil**				**Combined Average**			
Uni-Grapheme-TTS	4.87	2.07	40.65	9.63	4.89	2.01	28.88	8.37
Multi-Grapheme-TTS	5.15	2.09	43.31	9.92	4.99	2.02	30.36	8.41
G2P-TTS	5.16	2.09	44.27	10.38	4.91	2.02	30.07	8.54
Benchmark	5.06	2.08	43.67	10.04	4.79	2.01	30.54	7.98

Table 4: RMSE (frames per phone) between predicted and forced-aligned durations.

Models	Telugu	Hindi	Tamil
UGM	5.121	8.924	12.540
MGM	5.015	9.876	13.105
G2P	4.897	9.657	13.026
BMK	4.118	9.321	12.378

Table 5: Pearson correlation between predicted and forced-aligned durations.

Models	Telugu	Hindi	Tamil
UGM	0.787	0.525	0.618
MGM	0.807	0.533	0.624
G2P	0.818	0.564	0.657
BMK	0.866	0.692	0.695

Compared to Telugu, Hindi and Tamil show worse objective scores. For these two languages, punctuation marks were not retained in the corpus, which made pauses harder to predict. As a consequence, occasional pauses in the acoustics were frequently forced-aligned to non-pause phones, introducing errors in the reference durations. These unpredictable elongations inflated the objective measures, without perturbing the actual predictions too much. (Telugu, in contrast, used oracle pauses, inserted using Festvox's `ehmm` based on the acoustics.)

5.1.2. Acoustic Model

We used following four objective evaluations to assess the performance of the proposed methods in comparison to the benchmark system.

- **MCD:** Mel-Cepstral Distortion (MCD) to measure MCC prediction performance.
- **BAP:** to measure distortion of BAPs
- **F0 RMSE:** Root Mean Squared Error (RMSE) to measure the accuracy of F0 prediction. The error value was calculated on a linear scale instead of log-scale which was used to model the F0 values.
- **V/UV:** to measure voiced/unvoiced error.

In all these metrics, a lower value gives the better performance. While the objective metrics do not map directly to perceptual quality, they are often useful for system tuning. Table 3 presents the results on test data. As expected, the benchmark model performs well on most metrics. While the G2P Model performs well on Telugu and Hindi, the Uni-Grapheme model does well on Tamil. Overall, the proposed approaches compare favourably with the benchmark.

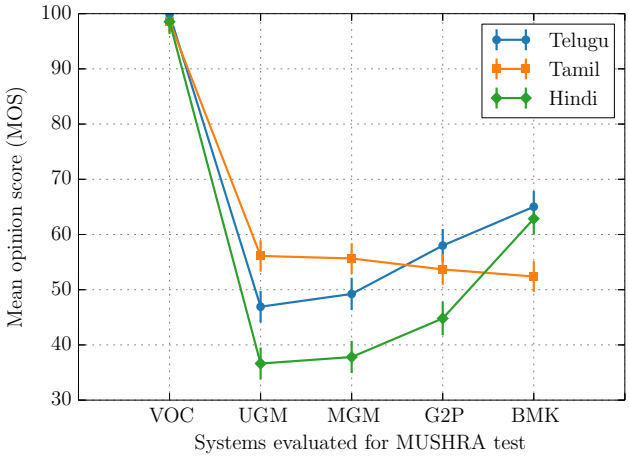

Figure 2: Performance of systems evaluated in the MUSHRA test for all three languages.

5.2. Subjective Evaluations

Three MUSHRA (MUltiple Stimuli with Hidden Reference and Anchor)[5] [26] tests were conducted to assess the naturalness of the synthesized speech. For each language, 16 native listeners were exposed to 20 sentences, chosen randomly from the test set. For each sentence, 5 unlabelled stimuli were presented in parallel: one for each of the four synthesis systems speaking that sentence plus copy-synthesis speech (i.e., vocoded speech, labelled as **VOC**) used as the hidden reference. Listeners were asked to rate each stimulus from 0 (extremely bad for naturalness) to 100 (same naturalness as the reference speech), and also instructed to give exactly one of the 5 stimuli in every set a rating of 100.

For Telugu and Hindi, we had access to a trusted pool of native speakers from IIIT-Hyderabad, while for Tamil we recruited paid workers via Amazon Mechanical Turk as listeners. The Mean Opinion Scores (MOS) from the tests are presented in Figure 2 with their standard deviation represented in log-scale. The benchmark model achieves a higher MOS in Telugu and Hindi, as expected, while in Tamil the Uni-Grapheme model

[5] `https://github.com/HSU-ANT/beaqlejs`

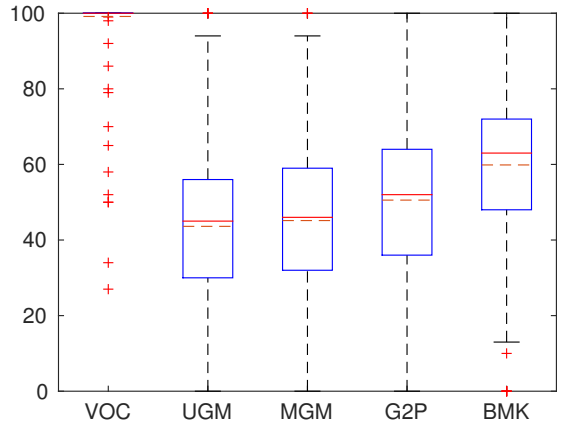

Figure 3: Box plot of absolute values from all three languages' listening tests. Red lines are medians, dashed lines means. Box edges show quartiles. Plus signs indicate outliers.

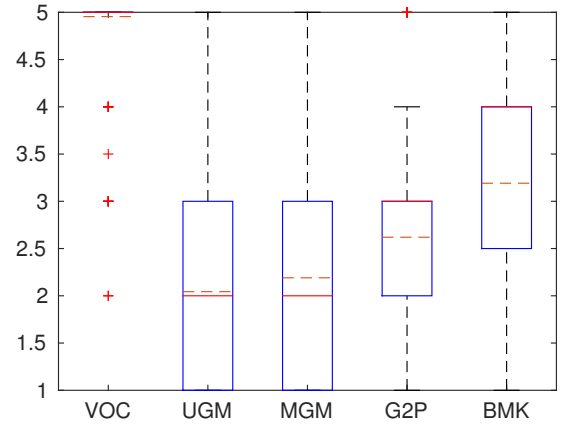

Figure 4: Box plot of aggregate ranks from listening tests (higher is better). Red lines are medians, dashed lines means. Box edges show quartiles. Plus signs indicate outliers.

achieves best performance. However, according to paired t-tests with Holm-Bonferroni correction for multiple comparisons, the difference with next best system is significant only in Telugu and Hindi. Among the proposed approaches, G2P performed significantly better than other two in Telugu and Hindi. However in Tamil, both G2P and benchmark performed worse than the rest. This strange behaviour can be attributed to two reasons: 1) the absence of a mechanism for detecting outliers in turker judgements (as opposed to the use of trusted pool of listeners for Hindi and Telugu); 2) the lack of our expertize in enhancing letter to sound rules specific to Tamil. The difference in ratings suggest that some additional rules or fine-tuning of lexicon may be required for Tamil.

The MUSHRA scores combined across all three languages for each system are presented in Fig. 3. For further analysis, each set of fifteen parallel listener scores was converted to ranks from 1 (worst) to 5 (best), with tied ranks set to the mean of the tied position. A box plot of these rank scores aggregated across all sentences and listeners is shown in Figure 4. Listener preferences between systems are also illustrated in Figure 5. All these figures indicate, G2P performed the best among the proposed approaches.

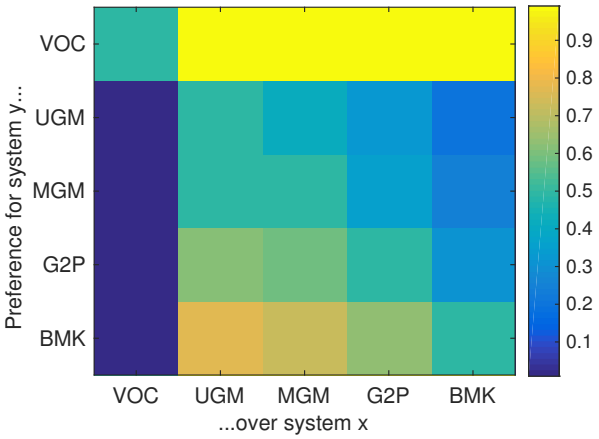

Figure 5: Preferences among systems (how often y was rated above x).

An interesting issue is that some test sentences include English-language words (e.g.: road, page, congress) due to frequent code-switching among the native speakers (also reflected in the text corpus). This affected the performance of G2P conversion for those sentences, in turn creating a marginal difference between G2P and benchmark over the listening test. G2P trained on large corpora of parallel text may remove such errors in the future, thereby improving the synthesis quality and reducing the gap towards the benchmark. [27] is one such recent attempt for synthesizing speech from code-mixed text.

No intelligibility evaluation was conducted since transcription word error rate (WER) has been found to be a poor metric for Indian languages, cf. [6]. However, we believe listeners do take into account intelligibility while rating the stimuli, even though they were asked to rate the naturalness.

6. Applications

The grapheme-to-phoneme conversion described herein enabled us to build indic-search[6], a search engine that helps end-users use ASCII to search for pages written in Unicode. Text-to-speech interfaces with ASCII input also enable users to type in their own pronunciation rather than conforming to a specific notation.

7. Conclusions

In this paper, we considered the problem of synthesizing speech from ASCII transliterated text of Indian languages. Our proposed approach first converts ASCII text to phonetic script, and then learns a DNN to synthesize speech from the phonetic script. We experimented with three approaches, which vary in the degree of manual supervision in defining phonemes. Our results show that G2P model with few assumptions is competitive with manually-defined phoneme models. All the data, and samples used in the listening tests are available online at: `http://srikanthr.in/indic-speech-synthesis`.

[6] `http://srikanthr.in/indic-search`

Acknowledgements: Thanks to Nivedita Chennupati and Spandana Gella for their contribution in data collection with Amazon Mechanical Turk. Also, thanks to Sivanada Achanta for evaluating the systems through listening tests. We thank Gustav Henter for proofreading. However, the errors that remain are the authors' responsibilities.

8. References

[1] A. N. S. Institute, "7-bit american standard code for information interchange," *ANSI X3*, vol. 4, 1986.

[2] U. Z. Ahmed, K. Bali, M. Choudhury, and S. VB, "Challenges in designing input method editors for indian lan-guages: The role of word-origin and context," in *Proceedings of the Workshop on Advances in Text Input Methods (WTIM 2011)*. Chiang Mai, Thailand: Asian Federation of Natural Language Processing, November 2011, pp. 1–9. [Online]. Available: http://www.aclweb.org/anthology/W11-3501

[3] R. S. Roy, M. Choudhury, P. Majumder, and K. Agarwal, "Overview of the fire 2013 track on transliterated search," in *Proceedings of the 5th 2013 Forum on Information Retrieval Evaluation*, ser. FIRE '13. New York, NY, USA: ACM, 2007, pp. 4:1–4:7. [Online]. Available: http://doi.acm.org/10.1145/2701336.2701636

[4] P. Gupta, K. Bali, R. E. Banchs, M. Choudhury, and P. Rosso, "Query expansion for mixed-script information retrieval," in *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, ser. SIGIR '14. New York, NY, USA: ACM, 2014, pp. 677–686. [Online]. Available: http://doi.acm.org/10.1145/2600428.2609622

[5] Y. Vyas, S. Gella, J. Sharma, K. Bali, and M. Choudhury, "POS tagging of English-Hindi code-mixed social media content," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, October 2014, pp. 974–979. [Online]. Available: http://www.aclweb.org/anthology/D14-1105

[6] K. Prahallad, A. Vadapalli, S. Kesiraju, H. Murthy, S. Lata, T. Nagarajan, M. Prasanna, H. Patil, A. Sao, S. King *et al.*, "The blizzard challenge 2014," in *Proc. Blizzard Challenge workshop*, 2014.

[7] K. Prahallad, A. Vadapalli, N. Elluru, G. Mantena, B. Pulugundla, P. Bhaskararao, H. Murthy, S. King, V. Karaiskos, and A. Black, "The blizzard challenge 2013–indian language task," in *Proc. Blizzard Challenge Workshop*, 2013.

[8] R. B, S. L. Christina, G. A. Rachel, S. Solomi V, M. K. Nandwana, A. Prakash, A. S. S, R. Krishnan, S. K. Prahalad, K. Samudravijaya, P. Vijayalakshmi, T. Nagarajan, and H. Murthy, "A common attribute based unified hts framework for speech synthesis in indian languages," in *8th ISCA Workshop on Speech Synthesis*, Barcelona, Spain, August 2013, pp. 311–316.

[9] P. Lavanya, P. Kishore, and G. T. Madhavi, "A simple approach for building transliteration editors for indian languages," *Journal of Zhejiang University Science A*, vol. 6, no. 11, pp. 1354–1361, 2005.

[10] G. Madhavi, B. Mini, N. Balakrishnan, and R. Raj, "Om: One tool for many (indian) languages," *Journal of Zhejiang University Science A*, vol. 6, no. 11, pp. 1348–1353, 2005.

[11] R. Gupta, P. Goyal, and S. Diwakar, "Transliteration among indian languages using wx notation." in *Proc. of KONVENS*, 2010, pp. 147–150.

[12] H. Li, A. Kumaran, V. Pervouchine, and M. Zhang, "Report of news 2009 machine transliteration shared task," in *Proceedings of the 2009 Named Entities Workshop: Shared Task on Transliteration*, ser. NEWS '09. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 1–18. [Online]. Available: http://dl.acm.org/citation.cfm?id=1699705.1699707

[13] E. V. Raghavendra, S. Desai, B. Yegnanarayana, A. W. Black, and K. Prahallad, "Global syllable set for building speech synthesis in indian languages," in *2008 IEEE Spoken Language Technology Workshop, SLT 2008, Goa, India, December 15-19, 2008*, 2008, pp. 49–52. [Online]. Available: http://dx.doi.org/10.1109/SLT.2008.4777837

[14] S. Kishore, R. Kumar, and R. Sangal, "A data driven synthesis approach for indian languages using syllable as basic unit," in *Proceedings of Intl. Conf. on NLP (ICON)*, 2002, pp. 311–316.

[15] H. Patil, T. Patel, N. Shah, H. Sailor, R. Krishnan, G. Kasthuri, T. Nagarajan, L. Christina, N. Kumar, V. Raghavendra, S. Kishore, S. Prasanna, N. Adiga, S. Singh, K. Anand, P. Kumar, B. Singh, S. Binil Kumar, T. Bhadran, T. Sajini, A. Saha, T. Basu, K. Rao, N. Narendra, A. Sao, R. Kumar, P. Talukdar, P. Acharyaa, S. Chandra, S. Lata, and H. Murthy, "A syllable-based framework for unit selection synthesis in 13 indian languages," in *Oriental COCOSDA held jointly with 2013 Conference on Asian Spoken Language Research and Evaluation (O-COCOSDA/CASLRE), 2013 International Conference*, Nov 2013, pp. 1–8.

[16] W. Oliver, "Unsupervised learning for text-to-speech synthesis," *Ph.D. dissertation, University of Edinburgh*, 2012.

[17] S. Sitaram, S. Palkar, Y. Chen, A. Parlikar, and A. W. Black, "Bootstrapping text-to-speech for speech processing in languages without an orthography," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, 2013, pp. 7992–7996. [Online]. Available: http://dx.doi.org/10.1109/ICASSP.2013.6639221

[18] O. Watts, S. Ronanki, Z. Wu, T. Raitio, and A. Suni, "The NST–GlottHMM entry to the Blizzard Challenge 2015," in *Proc. Blizzard Challenge Workshop*, 2015.

[19] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *Signal Processing Magazine, IEEE*, vol. 29, no. 6, pp. 82–97, Nov 2012.

[20] H. Zen, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *Proc. ICASSP*, 2013, pp. 7962–7966.

[21] Z. Wu, C. Valentini-Botinhao, O. Watts, and S. King, "Deep neural networks employing multi-task learning and stacked bottleneck features for speech synthesis," in *Proc. ICASSP*, 2015.

[22] H. Zen, K. Tokuda, and A. W. Black, "Statistical parametric speech synthesis," *Speech Communication*, vol. 51, no. 11, pp. 1039–1064, 2009.

[23] A. A. Raj, T. Sarkar, S. C. Pammi, S. Yuvaraj, M. Bansal, K. Prahallad, and A. W. Black, "Text processing for text-to-speech systems in indian languages." in *In Proc. of SSW6*, 2007, pp. 188–193.

[24] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Communication*, vol. 50, no. 5, pp. 434 – 451, 2008. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167639308000046

[25] H. Kawahara, M. Morise, T. Takahashi, R. Nisimura, T. Irino, and H. Banno, "Tandem-straight: A temporally stable power spectral representation for periodic signals and applications to interference-free spectrum, f0, and aperiodicity estimation," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, March 2008, pp. 3933–3936.

[26] S. Kraft and U. Zlzer, *BeaqleJS: HTML5 and JavaScript based Framework for the Subjective Evaluation of Audio Quality*, Linux Audio Conference, Karlsruhe, Germany, 2014.

[27] S. Sitaram and A. W. Black, "Speech Synthesis of Code Mixed Text," in *Proc. LREC*, 2016, pp. 3422–3428.