The 6th Intl. Workshop on Spoken Language Technologies for Under-Resourced Languages
29-31 August 2018, Gurugram, India

Brahmic Schwa-Deletion with Neural Classifiers: Experiments with Bengali

Cibu Johny, Martin Jansche

Google AI, London, United Kingdom

{cibu, mjansche}@google.com

Abstract

The Brahmic family of writing systems is an alpha-syllabary, in which a consonant letter without an explicit vowel marker can be ambiguous: it can either represent a consonant phoneme or a CV syllable with an inherent vowel ("schwa"). The schwa-deletion ambiguity must be resolved when converting from text to an accurate phonemic representation, particularly for text-to-speech synthesis. We situate the problem of Bengali schwa-deletion in the larger context of grapheme-to-phoneme conversion for Brahmic scripts and solve it using neural network classifiers with graphemic features that are independent of the script and the language. Classifier training is implemented using TensorFlow and related tools. We analyze the impact of both training data size and trained model size, as these represent real-life data collection and system deployment constraints. Our method achieves high accuracy for Bengali and is applicable to other languages written with Brahmic scripts.

Index Terms: Schwa-deletion, grapheme-to-phoneme, Bengali, Brahmic scripts, phonology, TensorFlow, neural networks

1. Introduction

The English word *table* exists as a loanword in Bengali, where it is pronounced /ʈebil/[1]. In Bengali script – also known as Eastern Nagari, a prominent member of the Brahmic family [1] – this is written as টেবিল. It is divided into three parts, one of which is টে followed by বি and ল. The first, টে, represents the syllable /ʈe/ and consists of the consonant letter ট (indicating /ʈ/) and the modifier ে appearing on its left and indicating /e/. This illustrates the workings of an alpha-syllabary [2] or *abugida*: in this type of writing system, consonant letters like ট and ব can support vowel modifiers, which attach to them. A typical visual cluster represents a consonant-vowel (CV) syllable: টে is read /ʈe/ and বি is read /bi/.

This raises two related questions: How should a consonant sound be written when it is not followed by a vowel? And how should a consonant letter be read if no modifier is attached?

In an abugida an unmodified consonant is thought to carry an *inherent vowel* and is read as a CV syllable. The precise phonetic quality of that vowel varies across languages, and sometimes within the same language. Following established conventions, we refer to the inherent vowel as *schwa*. We use that term in a purely graphemic sense, without any phonetic implications.

The English word *ball* is pronounced /bɔl/ as a loanword in Bengali. Its written representation বল consists of two unmodified consonant letters already familiar from above. The default reading of the inherent vowel in Bengali is /ɔ/, and so plain ব is read /bɔ/. We say that schwa has been phonetically realized.

In both বল /bɔl/ and the earlier example টেবিল /ʈebil/ the last letter ল has no modifiers attached and represents the final consonant sound. The inherent vowel of ল is not realized phonetically. We say that schwa has been deleted.

[1] Phonetic transcriptions are written with IPA symbols inside slashes.

Table 1: *The words* car *and* bus *as loanwords in various languages, shown in native orthography and transliteration*

Group	Languages	Orth.	Trans.	Orth.	Trans.
1	Bengali	কার	kār@	বাস	bās@
	Hindi, Marathi, Nepali	कार	kār@	बस	b@s@
	Gujarati	કાર	kār@	બસ	b@s@
2	Sinhala	කාර්	kār	බස්	b@s
	Malayalam	കാർ	kār	ബസ്	b@s
	Tamil	கார்	kār	பஸ்	p@s

The phenomenon of schwa deletion arises in several languages written in Brahmic scripts, including in Hindi [3, 4], Punjabi [5] in Brahmic Gurmukhi (as opposed to Perso-Arabic Shahmukhi) script, Marathi [6, 7], and others. Choudhury et al. [8] approach the problem diachronically for several languages, including Bengali.

Schwa deletion does not affect the entire Brahmic family. To show the absence of an inherent vowel, most Brahmic scripts can make use of ligatures and of an explicit vowel suppression mark, generically called *virama* following Sanskrit/Unicode conventions. One subset of languages – including Sanskrit, Sinhala, and the major Dravidian languages – requires explicit signaling of schwa deletion: a consonant without an inherent vowel must appear in ligated form or carry a visible virama; otherwise the inherent vowel is always realized.

The cross-linguistic situation is illustrated in Table 1. For easy comparison a uniform alphabetic transliteration (based on [9] and purely graphemic) is shown alongside the native abugida orthographies. Bengali and other languages in Group 1 do not explicitly suppress the final schwa: phonetically both *car* and *bus* end in a consonant, but this fact is not indicated in the orthography and must be inferred. Hindi ⟨kār@⟩[2], and ⟨b@s@⟩ contrast with Sinhala ⟨kār⟩ and ⟨b@s⟩. Sinhala and the Dravidian languages in Group 2 consistently use a visible virama (sometimes ligated in Malayalam) on the last letter of each word. Group 2 has a straightforward answer to our first questions: How should a consonant sound be written when it is not followed by a vowel? The consonant letter must be explicitly marked as lacking a vowel.

Such simple conventions are absent from Group 1 languages, not only those listed in Table 1 but also Punjabi and Kashmiri in Brahmic scripts, Bhojpuri, Maithili, Assamese, Sylheti, and others. Consequently Group 1 has no straightforward answer to our second question: How should a consonant letter be read if no modifier is attached? As the words बस ⟨b@s@⟩ /bəs/ in Hindi and বল ⟨b@l@⟩ /bɔl/ in Bengali illustrate, the inherent vowel can either be realized or deleted. This paper describes a general method for resolving this ambiguity.

[2] Brahmic graphemes are transliterated to the Latin script inside ⟨⟩. The inherent vowel is represented by ⟨@⟩ unless it has been explicitly suppressed, in which case no corresponding vowel symbol is written.

10.21437/SLTU.2018-54

Table 2: *Illustration of g2p stages and schwa outcomes*

	Representation stage *processing step*	Example					
1	Orthography	ক	ম	ল	ন	গ	র
	↓ *transliterate*						
2	Abstract graphemes, with schwa	k @	m @	l @	n @	g @	r @
	↓ *resolve schwa*						
3	Abstract graphemes, no schwa	k a	m a	l	n a	g a	r
	↓ *convert to phonemes*						
4	Phonemes	k ɔ	m o	l	n ɔ	g o	r

2. Background

The task of relating a written representation of a word to a phonemic representation is known as *grapheme-to-phoneme* conversion, or *g2p* for short. It is a common building block in several larger tasks, including transliteration [7], speech recognition, and text-to-speech synthesis (TTS) [3, 4], where the need for highly accurate pronunciations is especially acute.

2.1. Grapheme-to-phoneme conversion for Brahmic

The wide variety of Brahmic scripts used for writing South Asian languages creates both complexities and opportunities. The sheer diversity of scripts is an obstacle, but their similar workings provide an opportunity for applying a single generic solution to many similar problems. Our larger aim – beyond the scope of this paper – is towards generic grapheme-to-phoneme for Brahmic across similar languages, with minimal language-specific customization. A generic approach must arguably be able to deal with the schwa deletion phenomenon, as it affects a large group of languages to various degrees.

We conceive of schwa resolution as one module in a multistage process outlined in Table 2 which converts script characters (Stage 1) to phonemes (Stage 4). Table 2 illustrates the processing stages and steps using the Bengali place name কমলনগর (Kamalnagar), pronounced /kɔmolnɔgor/.

The first processing step – shared with a large body of related work on South Asian languages – is to transliterate the native script to an abstract grapheme representation (Stage 2). We use a modified version of ISO Romanization [9], solving two problems: First, it reduces complexity by mapping many different scripts (Devanagari, Eastern Nagari, Gurmukhi, etc.) into a single representation. Second, by mapping into Latin characters (as opposed to Devanagari, for example) the abstract grapheme representation (Stage 2 and 3) becomes alphabetic, matching the "size" of the alphabetic phoneme representation (Stage 4). In other words, the first processing step converts many different abugidas into a single alphabet.

The second processing step resolves each schwa grapheme ⟨@⟩ by either deleting it or replacing it with the grapheme ⟨a⟩. In the example in Table 2, we can see that the first and second instance of schwa become ⟨a⟩, while the third one gets deleted. As this step operates on the abstract graphemes, it allows for the possibility to make it language- and script-agnostic. The rest of this paper describes this step in detail.

The third processing step converts from abstract graphemes to phonemes. Even when large portions of the phoneme inventories may be shared among languages, this step is necessarily language-specific. As mentioned earlier, the precise phonetic details of the inherent vowel depend on the language. In the Bengali example in Table 2, some realized schwas surface as /ɔ/, while others surface as /o/. This is largely a consequence of larger g2p and phonological regularities in Bengali.[3] For example the raising of /ɔ/ to /o/ generally affects all occurrences of /ɔ/, regardless of whether they arose from an inherent vowel or from the independent vowel letter অ ⟨a⟩. It also parallels the raising of /ɛ/ to /e/ in Bengali. In the context of Bengali TTS [11] our preferred approach is to train sequence-to-sequence models on annotated data. For other languages, hand-written rewrite rules go a long way. Details are beyond the scope of the current paper.

Table 3: *Comparison of schwa deletion in Hindi and Bengali*

Language	Orthography	Transliteration	Phonemes
Hindi	राम	rām@	ram
Bengali	রাম	rām@	ram
Hindi	आठ सौ	āṭ@ sau	aʈsɔ
Bengali	আটশ	āṭ@ś@	aʈʃo
Hindi	कर्म	k@rm@	kərm
Bengali	কর্ম	k@rm@	kɔrmo
Hindi	अपर्याप्त	ap@ryāpt@	əpərjapt
Bengali	অপর্যাপ্ত	ap@ryāpt@	ɔpɔrdʒapto

2.2. Schwa deletion in Bengali compared with Hindi

Schwa deletion is a well-studied phenomenon of Hindi [3, 4]. Bengali shares some similarities, but differs in important ways.

Table 3 compares cognates between Hindi and Bengali. Words like the name *Ram* that end in simple final consonants are written and pronounced essentially the same in Hindi and Bengali: the final schwa deletes. On the other hand, in the compound word আটশ ⟨āṭ@ś@⟩ (*eight hundred*) the middle schwa deletes and the final schwa is pronounced.

Hindi has a very strong prohibition against final schwa. While this is a general preference in Bengali as well, a similarly strong restriction does not exist there. In fact, in several situations a final schwa must be pronounced in Bengali, when it would be deleted in the corresponding Hindi word. One large and systematic class of exceptions are those that end in consonant clusters in Hindi and in the corresponding cluster followed by /o/ in Bengali. Table 3 lists several examples.

Another systematic class of contexts where schwa deletion is difficult in Bengali are certain inflected verb forms. Roughly speaking, Bengali has several classes of consonant-stem verbs. The verb বল- /bɔl-/ (*to speak*) has inflected forms বলা /bɔla/, বলি /boli/, বলে /bɔle/ and /bole/[4], etc. The inflected form /bɔlo/ is often written as বল ⟨b@l@⟩, i.e. the final schwa must be pronounced. Deleting the final schwa in বল and pronouncing it as /bɔl/ changes the meaning to *ball*, an accidental homograph. The inflected form /bolte/ is written বলতে ⟨b@l@te⟩. Writing it instead with a ligated form as বল্তে ⟨b@lte⟩ would be highly unusual as that would obscure the shape of the stem বল-. To summarize, in the verb form ⟨b@l@⟩ the inherent vowel in ⟨l@⟩ must be pronounced; but in the related form ⟨b@l@te⟩ the inherent vowel in ⟨l@⟩ must be deleted. The latter situation is similar to Hindi बोलते ⟨bol@te⟩, but the occurrence of /-o/ in verb endings in Bengali has no analogue in Hindi that would give rise to similar problems there.

Crucially, Bengali has several homographs related to schwa

[3] The phonemic ambiguity of the realized schwa is arguably the next most important problem in Bengali g2p, second only to schwa deletion. This problem is not restricted to Bengali either. It also features prominently in Sinhala [10], where the inherent vowel is ambiguous between /a/ and /ə/. This also shows that the problem of phoneme conversion is independent of schwa deletion, as Sinhala lacks schwa deletion per the discussion around Table 1.

[4] A systematic class of frequent homographs, noted in passing.

Table 4: *Entries from the aligned Bengali pronunciation lexicon*

Orthography	Graphemes Phonemes						
অকথ্য	a ɔ	k k	@ o	t^{h} y̆ t t^{h}	@ o		
অকপট	a ɔ	k k	@ o	p p	@ ɔ	ṭ ṭ	@

deletion that can only be resolved contextually. We saw that বল ⟨b@l@⟩ is a homograph (/bɔlo/ vs. /bɔl/), and so are other frequent words like হল ⟨h@l@⟩ (/holo/ vs. /hɔl/) or কোন ⟨kon@⟩ (/kono/ vs. /kon/). Without contextual homograph disambiguation, schwa deletion is inherently ambiguous.

2.3. Related work

The schwa deletion problem can be viewed as a simple binary classification task: For each occurrence of the grapheme ⟨@⟩, decide whether it should be deleted or not. As in the existing literature, we evaluate quality in terms of classification accuracy.

The algorithm by Narasimhan et al. [3] uses morphological analysis with finite state transducers and cost models. It achieves 89% accuracy for Hindi schwa deletion.

The Epitran tool [12] adopts the same approach for Bengali. It uses the following regular expression contexts:

1. Schwa preceding a vowel (_V).
2. Schwa at the end of a word, following a vowel and consonants sequence (VC+_$).
3. Schwa sandwiched between vowel + consonant and consonant + vowel (VC_CV).

Chaudhary et al. [8] employ a diachronically motivated approach for Hindi schwa deletion that uses hand-written regular expression constraints. It achieves an accuracy of 96% for Hindi. With additional word-morphology information, this increases up to 99.9%.

Tyson and Nagar [4] construct an algorithm that uses syllable structure and stress assignment for words of two and three syllables for schwa deletion in Hindi, with an accuracy of 94%.

All of the above algorithms require significant linguistic knowledge. They use regular expressions that describe the triggering environments for schwa deletion, they use curated consonant cluster datasets, require syllabification, morphological analysis, and prosodic modeling. Moreover, aside from [8], all approaches are highly language-specific.

3. Method, data, and evaluation

Our approach is knowledge-lean and data-intensive. We train off-the-shelf neural network classifiers on data derived from a Bengali pronunciation lexicon, optimize hyper-parameters on a validation set, and evaluate on a held-out test set.

We started with a Bengali pronunciation lexicon[5] that had been built for use in a TTS system [11]. It contains more than 60,000 entries in Bengali script. Each entry consists of an orthographic string and a corresponding phoneme string that had been transcribed by native-speaker experts.

We transliterated the Bengali spellings into the abstract graphemic representation described above. We then force-aligned the abstract graphemes and phonemes for each entry based on the parametric approach described in [13]. This resulted in an aligned dataset over pairs of grapheme chunks and corresponding phoneme chunks. Table 4 shows a few entries from the aligned lexicon.

[5] https://github.com/googlei18n/language-resources/tree/master/bn/data

Table 5: *Example input feature columns*

L3	L2	L1	C	R1	R2	R3	R4	Label
BOS	a	k	@	p	@	ṭ	@	False
k	@	p	@	ṭ	@	EOS		False
p	@	ṭ	@	EOS				True

We used the alignment of the inherent vowel grapheme ⟨@⟩ to derive supervised labels for the neural classifiers. If ⟨@⟩ was aligned with a phoneme chunk, we assigned a label False; otherwise ⟨a⟩ was aligned with the empty string and we assigned a label True. This resulted in a labeled dataset with 84k schwa deletion/conversion instances.

We then prepared the following featurized dataset for the binary classifiers. The input feature columns are graphemes and represented as category columns with the entire grapheme set as the vocabulary. The target label is a Boolean indicating whether schwa is deleted or not. Crucially, no phonetic information from the pronunciation lexicon appears in the input features. Table 5 shows 3 consecutive instances of schwa processing produced from the single word অকপট whose alignment appears in Table 4. In this example, the left context length is 3 and the right one 4; both lengths are tunable hyperparameters.

We experimented with two additional sets of computed feature columns, expecting that they could potentially enhance classification quality. The first set of Boolean valued columns encodes the regular expression contexts of Epitran discussed above. The second set of columns decomposes the graphemes into linguistic features. For example, the grapheme ⟨ā⟩ is expressed in terms of articulatory features of its typical realization as an open central unrounded vowel.

We employ a DNNClassifier from TensorFlow [14] with most of the network parameters tuned as described below. The result of training is a binary classifier model which can predict schwa deletion. Table 6 lists the hyperparameters used.

Our objective is to build a model with high accuracy while controlling the training dataset size and tracking the output model size. As training data can be expensive, we want to understand the impact of data size on accuracy. Model size is the disk footprint of the fully trained model. Low capacity devices like mobile phones can be sensitive to model size, which is also a proxy for runtime complexity. We want to understand the possible trade-offs between model size and classification accuracy.

To that end, we set various input dataset size and output model size cut-offs, then tune hyperparameters and train a classifier under those constraints. A subset S of the given input dataset size is sampled uniformly at random from the full 84k dataset. The remaining dataset K is held out to evaluate the accuracy of the fully trained model.

Google Vizier [15] tuning is run on the sampled dataset S to determine hyperparameter values optimized for accuracy. In order to incorporate model size constraints, if the model size is greater than the given cut-off, the accuracy is demoted by the square of the ratio between the cut-off size and the model size.

Accuracy measurements are based on 250 Vizier tuning iterations. In each iteration, the sampled dataset S is randomly partitioned into training (90%) and validation (10%) sets. Each training cycle takes from a few minutes to an hour depending on the hyperparameters selected. The tunable hyperparameters and their respective value ranges are as per Table 6.

Note that the number of neural network layers and size of each layer are tunable parameters. We found that 4 layers or fewer are sufficient for the problem we are addressing. For small model sizes, the number of nodes in a layer is also small.

Left and right grapheme contexts are varied in length from

Table 6: *Hyperparameters, their range and tuned values for 16kB sized model trained from* 2^{16} *data giving accuracy 92.53%*

Hyperparameter	Range	Tuned value
Left Context Length	0 – 12	4
Right Context Length	0 – 12	4
Linguistic Feature Columns	Boolean	False
Regex Feature Columns	Boolean	False
Batch size	10 – 10000	7215
Steps	10 – 30000	2863
Epochs	1 – 260	226
Learning rate	10^{-4} – 10	0.26
Nodes in a Hidden Layer	2 – 2000	2
Number of Hidden Layers	1 – 4	3
Dropout	0 – 0.99	0.06
Activation function	ReLU, ReLU6, CReLU, ELU, SeLU, SoftPlus, SoftSign, Sigmoid, Tanh	CReLU
Loss Reduction	Mean, Sum, Sum by non-zero weights, Sum over batch size, Sum over non-zero weights	Sum over batch size
Optimizer	Adadelta, Adagrad, Adam, Ftrl, Proximal GD, Proximal Adagrad, RMSProp, Momentum	Adagrad

0 to 12 for tuning. However, they reach their maximum values only when model sizes are practically unbounded.

After hyperparameter tuning, the entire sampled dataset S is used for training the DNNClassifier with the network parameters tuned in the previous tuning step. The held-out dataset K is used to measure the accuracy of this final trained model.

4. Results and discussion

Figure 1 shows parametric plots of classification error in terms of model size for a variety of dataset sizes. Figure 2 shows the minimal observed classification error as a function of dataset size (typically achieved with large models). We observe:

1. Without any constraints, accuracy caps at around 97.5%.
2. When evaluated for word-ending schwa alone, our maximally trained model predicts schwa-deletion with 98.2% accuracy and for the remaining set of cases this is 97.1%.
3. Each quadrupling of the dataset size provides a similar relative reduction in error that has yet to level off.
4. The model size vs. error chart has a knee at around 12kB and 91% accuracy (9% error).
5. For fixed smaller dataset sizes, models seem to saturate. Increasing the model size yields diminishing returns on small datasets.
6. For all dataset sizes, the maximum achievable accuracy is only reached for the largest models.
7. Model sizes below 16kB limit the achievable accuracy.
8. The additional computed columns did not improve the accuracy and were inconsequential in the learning process, except at very small dataset sizes.

From the available accuracy data points, we can call out the tuned hyperparameters situated at the knee point of the curve representing the maximum dataset. This means, for a large training data, the hyperparameters listed in Table 6 would provide maximal accuracy while keeping the model size small.

The observed accuracies are consistent with the notion that schwa deletion is a "wide" problem where the classifiers need to encounter and memorize many different input contexts during training. If that is correct, it would also mean that knowledge-heavy approaches are unlikely to result in drastic improvements.

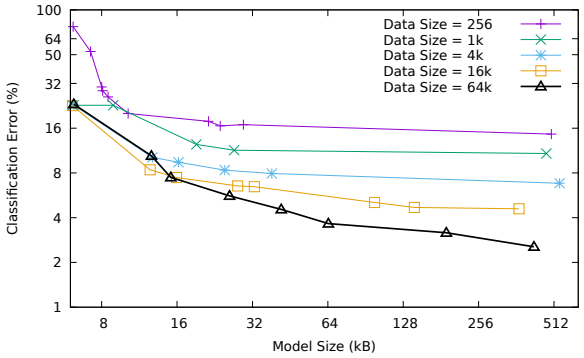

Figure 1: *Classification error by model size and data size*

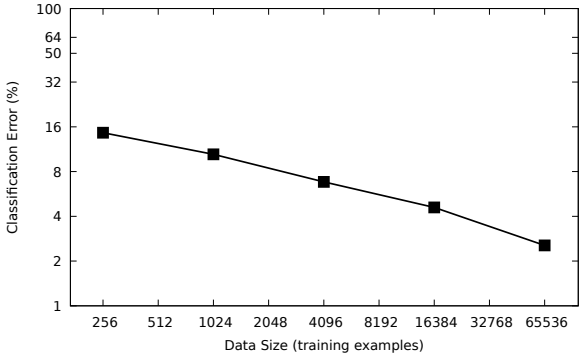

Figure 2: *Minimal classification error by data size*

5. Conclusion and future work

We have described a method for predicting schwa deletion with neural network classifiers. The classifiers are trained using only language- and script-agnostic features and supervised labels derived from an expert-transcribed pronunciation lexicon. All linguistic knowledge and all language-specific information comes exclusively from that lexicon.

When trained on as few as 256 examples, the classifiers achieve an accuracy of more than 85%. The model size can be kept compact by trading off various amounts of accuracy. With a large training set and without capping model size, an accuracy of more than 97% can be reached.

Our approach is generic and can be applied to other problems in Bengali or to other languages without major modifications. As the input feature set looks only at grapheme context, it is very general and can be used to specifically target other problems in pronunciation modeling. For example, we trained a neural classifier with our setup to predict the realization of ⟨a⟩ as /ɔ/ or /o/ with an accuracy of 92.5%. Another examples for such problems would be to predict the dental or alveolar articulation of the grapheme ⟨n⟩ in Malayalam, or to predict raising of mid vowels in Bengali.

Another avenue for exploration is multi-lingual training. Our approach makes it possible to train a single schwa deletion model on Bengali and e.g. Hindi data, using the target language as an input column. The hypothesis is that a large Bengali training set would allow us to achieve high accuracy on Hindi while requiring much fewer Hindi-specific training examples.

6. Acknowledgements

We thank Alexander Gutkin and Brian Roark for their helpful comments.

7. References

[1] P. T. Daniels and W. Bright, *The World's Writing Systems*. Oxford University Press, 1996.

[2] S. Nag and C. A. Perfetti, "Reading and writing: Insights from the alphasyllabaries of South and Southeast Asia," *Writing Systems Research*, vol. 6, no. 1, pp. 1–9, 2014.

[3] B. Narasimhan, R. Sproat, and G. Kiraz, "Schwa-deletion in Hindi text-to-speech synthesis," *International Journal of Speech Technology*, vol. 7, no. 4, pp. 319–333, 2004.

[4] N. R. Tyson and I. Nagar, "Prosodic rules for schwa-deletion in Hindi text-to-speech synthesis," *International Journal of Speech Technology*, vol. 12, no. 1, pp. 15–25, 2009.

[5] P. Singh and G. S. Lehal, "Punjabi text-to-speech synthesis system," in *Proceedings of COLING 2012: Demonstration Papers*, 2012, pp. 409–416.

[6] S. N. Kayte, M. Mundada, C. N. Kayte, and B. Gawali, "Grapheme-to-phoneme tools for the Marathi speech synthesis," *Int. Journal of Engineering Research and Applications*, vol. 5, no. 11, pp. 87–92, 2015.

[7] V. Ravishankar, "Finite-state back-transliteration for Marathi," *The Prague Bulletin of Mathematical Linguistics*, vol. 108, pp. 319–329, 2017.

[8] M. Choudhury, A. Basu, and S. Sarkar, "A diachronic approach for schwa deletion in Indo Aryan languages," in *Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology*, 2004, pp. 20–26.

[9] *International Standard ISO 15919: Transliteration of Devanagari and related Indic scripts into Latin characters*, 1st ed., 2001.

[10] A. Wasala, R. Weerasinghe, and K. Gamage, "Sinhala grapheme-to-phoneme conversion and rules for schwa epenthesis," in *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, 2006, pp. 890–897.

[11] A. Gutkin, L. Ha, M. Jansche, O. Kjartansson, K. Pipatsrisawat, and R. Sproat, "Building statistical parametric multi-speaker synthesis for Bangladeshi Bangla," in *5th Workshop on Spoken Language Technologies for Under-Resourced Languages (SLTU '16)*, 2016, pp. 194–200.

[12] D. R. Mortensen, S. Dalmia, and P. Littell, "Epitran: Precision G2P for many languages," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[13] M. Jansche, "Computer-aided quality assurance of an Icelandic pronunciation dictionary," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC '14)*, 2014, pp. 2111–2114.

[14] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "TensorFlow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*, 2016, pp. 265–283.

[15] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley, "Google Vizier: A service for black-box optimization," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1487–1495.