Universal Grapheme-based Speech Synthesis

Sunayana Sitaram[1], *Alok Parlikar*[1],
Gopala Krishna Anumanchipalli[2], *Alan W Black*[1]

[1]Carnegie Mellon University, USA
[2]University of California, San Francisco, USA

ssitaram@cs.cmu.edu, aup@cs.cmu.edu,
AnumanchipalliG@neurosurg.ucsf.edu, awb@cs.cmu.edu

Abstract

Grapheme-to-phoneme conversion follows the text processing step in speech synthesis. Typically, lexicons or Letter-to-Sound rules are used to map graphemes to phonemes. However, in some languages, such resources may not be readily available. In this paper, we describe a universal front end that supports using grapheme information alone to build usable speech synthesis systems. This work takes advantage of an explicit mapping of Unicode characters from a wide range of scripts to a single phoneset to create support for building speech synthesizers for most languages in the world. We compare the efficacy of this front end to the baseline approach of treating every single grapheme as a separate phoneme for synthesis by building voices for twelve languages across several language families and to front ends with linguistic knowledge in languages with higher resources. In addition, we improve our models by using Random Forests as opposed to using single Classification and Regression Trees. We find that the common universal front end performs better than the raw graphemes in general. We also find that using Random Forests lead to a significant improvement in synthesis quality, which is better than the quality of the knowledge based front end in many cases.

Index Terms: speech synthesis, lexicons, pronunciation, low resources

1. Introduction

Grapheme to Phoneme (g2p) conversion is an essential module in any Text to Speech (TTS) system. This process involves reading a string of words, represented by symbols in the script at hand, and mapping them into a set of phonetic symbols representing the pronunciation of the words. This stage occurs after the text processing stage, in which abbreviations, numbers and other symbols are expanded into words in the language.

Usually, g2p conversion is done by looking up lexicons developed for the language. In some languages, the relationship between graphemes and phonemes follows well defined rules, in which case hand-written g2p rules can be used. Even for those languages that have large lexicons available, learned g2p models or rules may be used to deal with unknown words. Language specific knowledge (morphological, phonotactic, sometimes syntactic) is necessary for building these resources, which makes developing a front end for a new language expensive and time consuming.

The g2p conversion problem varies from language to language. Some languages may require the development of large lexicons, while for others, mapping graphemes to phonemes may be trivial. Given a new language that we need to build a speech synthesizer for, we need to come up with g2p rules or find an appropriate lexicon for the language. Automatic, language independent methods of finding this mapping have been proposed[1] but they depend upon the availability of training data, such as hand-crafted lexicons which may not be available for low-resource languages.

One solution to this problem is to make the simplifying assumption that the orthography of the language is phonetic, and that each grapheme is a phoneme [2]. This allows us to build baseline voices for languages without knowing anything about the phonetics of the language. Another solution is to exploit resources that transliterate graphemes into phonemes. The Unicode specification [3] provides a standardized digital representation of scripts in most languages of the world. Qian et. al [4] have developed a toolkit that provides universal transliteration for characters in the Unicode specification.

In our speech synthesizers, we use individual Classification and Regression Trees (CART) to model the spectrum, pitch and duration. In low-resource scenarios, we typically do not have much data to train these trees on. Using Random Forests [5], which is an ensemble learning technique that uses multiple CART trees may provide advantages in such cases by splitting the data in different ways and making better use of features.

In this work, we took speech corpora from twelve languages from disparate language families and various scripts and built synthetic voices from them. We first built voices with the simplest possible front end: assume that each grapheme is a phoneme. Then, we made use of the universal mappings from Unicode characters to phonemes to build voices. Finally, we attempted to improve on these voices by making use of Random Forests in our models. A few of the languages we built voices for are high resource languages for which front ends with linguistic resources are available. We built voices with this available knowledge for these languages. We evaluated our systems objectively and subjectively to compare these different techniques.

The rest of the paper is organized as follows: Section 2 describes the languages and resources we used for our experiments. Section 3 describes the techniques we used to build our voices in more detail. The objective and subjective evaluation results are presented in sections 4 and 5 respectively. Section 6 concludes.

2. Languages and Resources

Since our goal is to use the Unicode based mapping to produce a universal grapheme-to-phoneme frontend, we ran experiments with twelve different languages. These languages were chosen to vary across writing systems, language families as well as the amounts of resources available for the language. We also used speech databases of different sizes in order to compare our techniques across different amounts of data.

Two of our languages, Hindi and Konkani are from the Indo-Aryan language family. Hindi is a major language in India with over 180 million native speakers. Hindi (Modern Standard Hindi) uses the Devanagari script as its standard writing system, and the script has a phonetic alphabet with a few rules to handle special cases. To build our synthetic voices, we used a corpus of Hindi from the IIIT-H Indic Databases[6]. Konkani is the official language of the Indian state of Goa, and is a minority language in a few other states. It has around 8 million native speakers and uses scripts such as Latin, Kannada, Malayalam and even Arabic. We used a corpus of Konkani from the CMU SPICE project[7] that used the Latin script.

Tamil is also an Indic language, from the Dravidian family of languages. It is spoken by over 70 million speakers in southern India as well as Sri Lanka, Singapore and Mauritius. We based our Tamil experiments on the IIIT-H Indic corpus[6].

Iraqi Arabic, Dari and Pashto are languages that all use the Arabic script in their written forms. Iraqi Arabic is the Arabic dialect native to the Mesopotamian basin of Iraq has about 15 million speakers. Dari is a dialect of Persian that is the standard language used in Afghanistan. It has about 18 million native speakers. Pashto is the other official language of Afghanistan, is the oldest preserved Iranian language and has over 40 million speakers. The corpora we used for Iraqi, Dari and Pashto were used for building synthesizers as part of the DARPA TRANSTAC project.

Russian is a Slavic language spoken by over 155 million native speakers and uses the Cyrillic script. Thai is a language spoken by over 20 million people and uses its own script. We collected our Russian and Thai speech corpora from the SPICE[7] dataset.

Inupiaq is a Inuit language spoken by about 2100 people in northern and north western Alaska, and it uses the Latin+ script as its orthography. Ojibwe, also known as Anishabnaabe is an indigenous language of the Algonquian family and is native to Canada and the United States. It has about 56000 native speakers and uses the latin script in its written form. Our corpus for Inupiaq and Ojibwe was collected as part of the Endangered Languages project at Carnegie Mellon University.

In the European family of languages, we did experiments with English and German. For English, we used the ARCTIC [8] recordings of the speaker SLT and for German, we used a similar corpus that we recorded locally. Table 1 shows the amount of speech data in minutes that we used to build synthetic voices for each of these languages.

As we can see in Table 1, the data for our languages varied from as little as 5 minutes of recorded speech for languages like Konkani and Inupiaq to over an hour of speech for English and Dari. In addition, the quality of recordings was also not uniform across the databases, with Russian being significantly worse than the other databases. We did this in order to mimic real-world scenarios of data availability for low-resource languages and to measure the gains our techniques would provide across different data sizes.

3. Experimental Setup

We performed our experiments in the context of the Festival[9] speech synthesis engine. We built statistical parametric synthesis models using the Clustergen[10] framework and used the Festvox[11] suite of tools to build Festival voices.

Table 1: *Data available for different languages*

Language	Duration (minutes)	Script
English	66	*Latin*
Dari	63	*Arabic*
Hindi	56	*Devanagari*
Iraqi	61	*Arabic*
Pashto	56	*Arabic*
German	53	*Latin*
Tamil	41	*Tamil*
Thai	25	*Thai*
Ojibwe	12	*Latin*
Russian	6	*Cyrillic*
Konkani	5	*Latin*
Inupiaq	5	*Latin*

For each language, we built up to four different voices: (i) Raw grapheme voice, where each (Unicode) character is treated as a phoneme, (ii) UniTran based voice, where we used the universal mapping from graphemes to strings of phonemes (iii) Random Forests (RF) voice, in which we used Random Forests instead of single CART trees in our models, using the UniTran mappings (iv) knowledge based voice, which used any knowledge we had for that language including lexical resources. For some of the languages, we could not build knowledge based voices and for them, we only present the comparison of the raw grapheme, UniTran and RF voices.

3.1. Raw Grapheme Synthesis

Given a new language to build a synthetic voice in, we have used raw graphemes for synthesis as the baseline method. The idea is simple: we take the text in the presented orthography. We then explode the text into individual Unicode code points, and then each of the Unicode characters becomes a phoneme for purposes of synthesis. Here, we treat each of these characters as a grapheme. This method is automatic and universal: the produced set of ”phonemes” are simply characters, and there is no information available about how they are pronounced, or which phonetic or phonotactic features they possess. Each of these phonemes (characters) is considered to have three states in the context of HMM synthesis within Clustergen.

The advantage with the raw grapheme method is that it can be used when we have absolutely no information about the phonetics of a language - all we need is speech data and corresponding transcripts in the orthography of the language. The disadvantage with this method is that the models cannot use phonetic feature information while clustering similar phonemes together, since we have no information about what the phonemes actually are. Another disadvantage is that multiple characters that may actually map to a single phoneme in the language (like vowel markers and vowels) now map to different phonemes, which may lead to less data and context for each phoneme.

3.2. Proposed Method with UniTran

UniTran[4] is a transliteration framework to convert UTF-8 encoded text into a guessed phonetic transcription in the WorldBet[12] or X-Sampa. It supports about 40 different character code pages in the Unicode specification. The two notable omissions are the Latin character set, and the Chinese Kanji.

UniTran is distributed with the Natural Language Toolkit[13] for Python.

We took the character mapping tables from UniTran and converted them for use in Festvox. While this conversion was mostly automatic, we had to deal with a few special cases. In some cases, UniTran specified alternate mappings for a single character. Festvox defaulted to just using the first provided mapping. In addition, we also added support for the Latin characters. While the Latin alphabet isn't necessarily phonetic, we made reasonable assumptions about which letter maps to which phoneme in the X-Sampa set. This will not be optimal for all languages but allowing some mapping into a symbol set with phonetic features allows for later models to make more complex distinctions that would not be possible if they were mapped to an unknown symbol.

We took all the phonemes that UniTran maps to, and created a master phoneset description file in Festvox that contains several features for each phone, such as whether it's a vowel or a consonant, for vowels: height, length, frontness and rounding; and for consonants: the type, voicing and place of articulation.

When we create a new Festvox voice, we go over the text corpus and map each Unicode character into its appropriate phone. We then accumulate the list of phones actually used in the language at hand and select a subset of the master phoneset above, to customize the phoneset in the particular language. This means phonemes that were not observed in the training data of our voice, but still exist in the target language will not have a phoneset entry. This is not a problem, because even if a phoneset entry existed, we wouldn't have training data to build parametric models for cepstra and prosody for the particular unseen phone.

Once we have the grapheme to phoneme mapping this way, we represent each phoneme with three HMM states and use standard Clustergen techniques to build the voice.

3.3. Synthesis with Linguistic Knowledge

Some of the languages we used in our task are high-resource languages, and there has been prior work on building synthetic voices for them. For English, German, Hindi, Iraqi and Tamil, we have developed front ends that have linguistic knowledge incorporated in them, beyond simple phonetic features. These are in some sense "oracle" languages for the task at hand.

For English and German, we used our standard Festival front ends. These front ends have a large lexicon and letter to sound rules trained from this lexicon, for unseen words. For Iraqi, we also used a large lexicon that specified pronunciation variants of words, since the diacritics that map to short vowels are not written in the script. We used the first pronunciation variant of a word when multiple variants were present. For Hindi and Tamil, we used g2p mappings for all the characters and added post-lexical rules for nasalized vowels, contextual nasal consonants. For Hindi, we added rules for terminal and medial schwa deletion. For Tamil, we added rules for contextual voicing of consonants. In all these cases our linguistically based voices require substantial linguistic and expert resources beyond the simple grapheme or UniTran based voices.

3.4. Synthesis with Random Forests

For our Random Forests voices, we started with the UniTran mapping for the grapheme to phoneme conversion. We applied our now standard random forest technique, originally developed for our standard voices. For these voices, we build 20 decision trees to predict spectral features, where each tree is randomly restricted to 70% of the standard prediction features.

Each tree individually will typically give worse results than a tree built with all features, but the combination of multiple trees built with different features will typically give a substantial improvement. Although this technique is not specifically designed for grapheme based voice builds, the combination of predictions from different trees allows better use of features and avoids over splitting the data, which we felt may be helpful in this low resource scenario.

4. Objective Evaluation

In order to compare the different grapheme to phoneme conversion strategies, we built full synthetic voices out of them. We held out 10% of the data during voice building. On this data, we compared the synthetic speech with reference recorded speech by looking at the Mean Mel Cepstral Distortion[14] (MCD) of the predicted cepstra. Since this is a distance measure, a lower value suggests better synthesis. Kominek [15] has suggested that MCD is linked to perceptual improvement in the intelligibility of synthesis, and that an improvement of about 0.08 is perceptually significant and an improvement of 0.12 is equivalent to doubling the data.

Let us first compare our UniTran based method to the baseline with raw graphemes, shown in Table 3. On eight languages in our set, we see that using the phonetic features yields an improvement in the synthesis quality. The voices for Dari and Pashto actually get worse. Our analysis shows that this is caused by the inherent vowel in the Arabic script. The basic UniTran mappings are appropriate for Iraqi (which is a dialect of Arabic), but not appropriate for Pashto and Dari that are in different language families, but use the Arabic script. On the Thai voice, the MCD goes up, but the pitch prediction error goes down from 35Hz to 30Hz. Ojibwe, that uses a Latin script, doesn't have a very different synthesis model when moving from raw graphemes to phonetic features with UniTran.

Let us now look at the comparison of our UniTran based voices to voices with linguistic knowledge. We see that across all five languages for which we have knowledge-based front ends, the voice with knowledge is much better than the UniTran based voice. The difference is much larger (0.32) on English, and relatively smaller (0.09) on Iraqi. This may be because the amount of linguistic knowledge that went into the English voice was much larger than that in the Iraqi voice. Adding higher level knowledge about syllable structure and morphology for these languages may further improve our techniques.

Table 2: *MCD for languages built with raw graphemes, UniTran and knowledge (when available)*

Language	Raw	UniTran	Knowledge Based
English	5.23	5.11	**4.79**
Dari	**4.78**	4.95	
Hindi	5.10	5.05	**4.94**
Iraqi	4.77	4.72	**4.63**
Pashto	**4.91**	4.96	
German	4.72	4.30	**4.15**
Tamil	5.10	5.04	**4.90**
Thai	**4.82**	4.98	
Ojibwe	6.72	6.71	
Russian	5.13	**4.78**	
Konkani	5.99	**5.87**	
Inupiaq	4.79	**4.68**	

Next, we will compare MCD for the UniTran and RF voices shown in Table 3. We see that in every single case, we get a significant improvement in MCD. In case of Hindi, Iraqi, German and Tamil we are able to perform better than the knowledge based front end without Random Forests. We expect that building a Random Forest voice using the knowledge based front end would lead to even better improvements, but for our low-resource scenario where such a front end will not be available, this result is very encouraging.

Table 3: *MCD for languages built with UniTran vs. Random Forests*

Language	UniTran	Random Forests
English	5.11	4.91
Dari	4.95	4.80
Hindi	5.05	4.88
Iraqi	4.72	4.56
Pashto	4.96	4.80
German	4.30	4.10
Tamil	5.04	4.85
Thai	4.98	4.74
Ojibwe	6.71	6.19
Russian	4.78	4.64
Konkani	5.87	5.59
Inupiaq	4.68	4.56

5. Subjective Evaluation

So far, we saw improvements in objective metrics both while going from raw graphemes to UniTran and from single CART trees to Random Forests. We compared these conditions in subjective evaluations for English, German, Russian, Hindi and Tamil. Our choice of languages was based on the availability of subjects for listening tests.

We used Testvox [16] to carry out all our subjective tests both locally and on Amazon Mechanical Turk. In order to ensure that native speakers took the test, we translated the instructions to the respective languages. Each participant listened to audio clips in random order and was asked to pick the one she preferred, with the option of picking ”no difference”. Each participant listened to either 5 or 10 pairs of clips, with most participants listening to 10 pairs.

Table 4 shows the results of the subjective listening comparison between raw graphemes and the Unitran-based systems. We can see that in all five languages, subjects preferred the UniTran voices to the Raw Graphemes voices. In some languages, this difference was higher while in languages like Hindi and Tamil where the UniTran method does not provide a huge gain over raw graphemes due to the nature of the writing system, the difference wasn’t as high. The quality of the Russian speech data being lower probably made it harder to differentiate between the two voices.

Table 5 shows the comparison between the Unitran-based systems with and without and Random Forests. There was a preference for the RF voices over the UniTran voices in all cases. Once again, the Russian speech seemed harder to differentiate.

So far, we saw that the subjective results mimicked the MCD trends that we saw earlier. Since our main goal was to see how far we could get with our best grapheme voices when compared to voices built with knowledge, we carried out transcription tests in English, German and Hindi. We asked 10 subjects to transcribe 10 sentences each in English, German and Hindi for the Knowledge Based and Random Forests grapheme conditions. Table 6 shows the percentage of words transcribed correctly for all the languages.

Table 4: *Raw graphemes vs. UniTran preference*

Language	Participants	Prefer RG	Prefer UniTran	No difference
English	13	22%	61%	17%
German	12	29%	54%	17%
Russian	11	32%	43%	25%
Hindi	12	38%	51%	11%
Tamil	12	28%	58%	14%

Table 5: *UniTran vs. Random Forests preference*

Language	Participants	Prefer UniTran	Prefer RF	No difference
English	12	31%	61%	8%
German	9	36%	51%	13%
Russian	12	29%	47%	24%
Hindi	12	31%	51%	18%
Tamil	12	37%	53%	10%

Table 6: *Words transcribed correctly for Knowledge Based and RF grapheme systems*

Language	Knowledge Based	Random Forests
English	87.14%	66.52%
German	90.85%	89.89%
Hindi	88.19%	86.34%

Here we see that for English, the grapheme RF voice is still significantly worse than the knowledge based one. This is due to two reasons - the knowledge that went into the English voice is substantially more than the other languages and the nature of the English script makes a grapheme-based technique quite inappropriate for it. However, we see that for both German and Hindi, the difference between the usability of the RF and knowledge based voices is very low, and all the voices are almost at 90% transcription accuracy.

6. Conclusions

In this paper, we introduced a method to build grapheme-based voices by using a universal mapping of Unicode characters to phonemes, which has now been incorporated into the standard distribution of the Festvox voice building tools. We built voices using this front end for twelve languages. We also built our standard Random Forest voices using the UniTran mapping. We showed that the UniTran voices are better than baseline raw grapheme voices. In addition, we also showed that using Random Forests as a modeling technique improves the UniTran based voices and in most cases, is better than the knowledge based voice modeled with single trees. We believe that this new frontend in Festvox will allow voices for new languages to be setup more easily and help speed up building synthetic voices for many more languages of the world. In addition, using Random Forests may help in low resource scenarios to make the best use of the available data.

7. References

[1] W. M. Daelemans and A. P. Van den Bosch, “Language-independent data-oriented grapheme-to-phoneme conversion,” in *Progress in speech synthesis*. Springer, 1997, pp. 77–89.

[2] G. Anumanchipalli, K. Prahallad, and A. Black, “Significance of early tagged contextual graphemes in grapheme based speech synthesis and recognition systems,” *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*, pp. 4645–4648, 2008.

[3] C. Unicode Staff, *The Unicode standard: worldwide character encoding*. Addison-Wesley Longman Publishing Co., Inc., 1991.

[4] T. Qian, K. Hollingshead, S.-y. Yoon, K.-y. Kim, R. Sproat, and M. LREC, “A python toolkit for universal transliteration.” in *LREC*, 2010.

[5] L. Breiman, “Random forests,” *Machine Learning*, vol. 45(1), pp. 5–32, 2001.

[6] K. Prahallad, E. N. Kumar, V. Keri, S. Rajendran, and A. W. Black, “The IIIT-H Indic Speech Databases,” in *Proceedings of Interspeech*, Portland, OR, USA, September 2012.

[7] T. Schultz, A. W. Black, S. Badaskar, M. Hornyak, and J. Kominek, “Spice: Web-based tools for rapid language adaptation in speech processing systems,” in *Eighth Annual Conference of the International Speech Communication Association*, 2007.

[8] J. Kominek and A. W. Black, “CMU Arctic Databases for Speech Synthesis,” in *Proceedings of the 5th Speech Synthesis Workshop*, Pittsburgh, Pennsylvania, June 2004, pp. 223–224.

[9] A. W. Black and P. Taylor, “The Festival Speech Synthesis System: system documentation,” Human Communication Research Centre, University of Edinburgh, Tech. Rep., January 1997. [Online]. Available: http://www.cstr.ed.ac.uk/projects/festival

[10] A. W. Black, “CLUSTERGEN: A Statistical Parametric Synthesizer using Trajectory Modeling,” in *Proceedings of Interspeech*, Pittsburgh, Pennsylvania, September 2006, pp. 194–197.

[11] A. W. Black and K. Lenzo. (2002) Building Voices in the Festival Speech Synthesis System. [Online]. Available: http://festvox.org/bsv

[12] J. L. Hieronymus, “Ascii phonetic symbols for the worlds languages: Worldbet,” *Journal of the International Phonetic Association*, vol. 23, 1993.

[13] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python*. ” O’Reilly Media, Inc.”, 2009.

[14] M. Mashimo, T. Toda, K. Shikano, and N. Campbell, “Evaluation of cross-language voice conversion based on gmm and straight,” 2001.

[15] J. Kominek, “TTS From Zero: Building Synthetic Voices for New Languages,” Ph.D. dissertation, Carnegie Mellon University, 2009.

[16] A. Parlikar, “Testvox: Web-based framework for subjective evaluation of speech synthesis,” *Opensource Software*, 2012.