The NST–GlottHMM entry to the Blizzard Challenge 2015

O. Watts[1]*, S. Ronanki*[1]*, Z. Wu*[1]*, T. Raitio*[2]*, A. Suni*[2,3]

[1]Centre for Speech Technology Research, University of Edinburgh, UK
[2]Dept. of Signal Processing and Acoustics, Aalto University, Finland
[3]Institute of Behavioural Sciences, University of Helsinki, Finland

owatts@inf.ed.ac.uk, s.ronanki@sms.ed.ac.uk zhizheng.wu@ed.ac.uk
tuomo.raitio@aalto.fi, Antti.Suni@helsinki.fi

Abstract

We describe the synthetic voices forming the joint entry into the 2015 Blizzard Challenge of the Natural Speech Technology consortium, Helsinki University, and Aalto University. The 2015 Blizzard Challenge presents an opportunity to test and benchmark some of the tools we have developed to address the problem of how to produce systems in arbitrary new languages with minimal annotated data and language-specific expertise on the part of the system builders. We here explain how our tools were used to address these problems on the different tasks of the challenge, and provide some discussion of the evaluation results. Some additions to the system used to build voices for the previous Challenge are described: acoustic modelling using deep neural networks with jointly-trained duration model, and an unsupervised approach for handling the phenomenon of inherent vowel deletion which occurs in 3 of the 6 target languages.

Index Terms: statistical parametric speech synthesis, unsupervised learning, vector space model, glottal inverse filtering, deep neural network, glottal flow pulse library, schwa-deletion

1. Introduction

We here describe the synthetic voices used to synthesise speech for the Natural Speech Technology[1] consortium, Helsinki University and Aalto University's joint submission to the 2015 Blizzard Challenge. We make use of tools and techniques developed for the Simple4All entries to the 2013 and 2014 Blizzard Challenges and further develop these. As with previous Challenges, the 2015 Challenge provides an opportunity to test and benchmark the technology we have been developing; the production of systems in arbitrary new languages with minimal language-specific expertise on the part of the system builders remains a central concern of our work.

Compared to our previous (Simple4All) entries, the text processing and speech parameterisation steps are largely unchanged. A notable exception is our attempt to detect schwa-deletion based on acoustic evidence: this new development is described in Section 2.1.2 below.

Last year, we introduced deep neural network (DNN) based modelling of voice source into our system, while the rest of the GlottHMM [1] parameters were generated using a standard HMM-based speech synthesis framework. HMM acoustic models have several known deficiencies: data fragmentation due to decision tree clustering, the assumption of independence between acoustic parameter streams where model clustering is performed separately for each stream, and the assumption of independence between coefficients within each stream where diagonal covariance matrices are used. These false assumptions affect complex parameterisations such as GlottHMM severely, as vocal tract and voice source spectrum are separately modelled with LSFs. In contrast, neural networks do not rely on these assumptions, and we have thus used DNNs for all of our acoustic modelling.

[1]http://www.natural-speech-technology.org

2. System Description

2.1. Text processing

The tools used for building TTS front-ends for our entry to this year's Challenge are an improved and extended version of those used to prepare our entry to the last two years' Challenges [2, 3]. The principal extension to our front-end is a module for automatically detecting and predicting schwa-deletion, which was used for three of the six target languages. Other parts of the toolkit are similar or identical, and so much of the following description very closely follows descriptions given in [2, 3].

The only language-specific input to our system is the data used for training. This consisted of the audio data for each language transcribed in plain orthography at the utterance level which was distributed for the Challenge: 2 hours for Bengali, Malayalam and Marathi and 4 hours for Hindi, Tamil and Telugu. In addition, we made use of large quantities of unannotated text data for building word representations in an unsupervised manner – this consisted of approximately 3.6, 15.9, 4.9, 1.9, 8.8 and 9.1 million tokens of text for Bengali, Hindi, Malayalam, Marathi, Tamil and Telugu, respectively, which we obtained from Wikipedia.

Text which is input to the system is assumed to be UTF-8 encoded: given UTF-8 text, text processing is fully automatic and makes use of a theoretically universal resource: the Unicode database. Unicode character properties are used to tokenise the text and characterise tokens as words, whitespace, punctuation etc. Our front-ends currently expect text without abbreviations, numerals, and symbols (e.g. for currency) which require expansion.

2.1.1. Naive alphabetisation

We used the same method as in the 2014 Challenge to create a naive alphabetisation of text input based on characters' names in the Unicode database [3]. Take for example the Hindi word, *prasid'dha*, meaning '*famous*':

प्रसिद्ध

The Unicode representation of this token consists of eight enti-

ties whose names are as follows:

- DEVANAGARI LETTER PA
- DEVANAGARI SIGN VIRAMA
- DEVANAGARI LETTER RA
- DEVANAGARI LETTER SA
- DEVANAGARI VOWEL SIGN I
- DEVANAGARI LETTER DA
- DEVANAGARI SIGN VIRAMA
- DEVANAGARI LETTER DA

This representation already abstracts away from the ligatures and variable ordering of the surface devanagari text. For example, the <*pra*> conjunct is represented by the sequence <*pa virama ra*> and the graphical right–left ordering of the elements <*sa*> and <*i*> is changed to the left–right direction of the rest of the text. The alphabetic representations used in these names (*PA*, *RA*, *SA* etc.) give a transcription which is used instead of the Unicode characters, but several simple rules which capture general knowledge about this family of scripts are used to modify the string of letters obtained. A *LETTER*'s inherent vowel is altered when a modifier follows, a modifier being any letter with *VOWEL SIGN* in its name, or *VIRAMA*, *ANUSVARA* or *CANDRABINDU*. Vowel signs are used to replace the preceding letter's inherent vowel, *VIRAMA* to delete it, and *ANUSVARA* and *CANDRABINDU* to nasalise it (which we represent by appending *m* to the modified vowel). Applying these rules to the above example yields the sequence *P R A S I D D A*.

2.1.2. Inherent vowel deletion

A well-known phenomenon of the Indo-Aryan language group – which includes three of our target languages, Bengali, Hindi and Marathi – is *schwa deletion*, where a character's inherent vowel is not pronounced, even though not suppressed by a vowel sign or virama [4]. For example, the final <a> of the Hindi word *prasid'dha*, mentioned above, is not pronounced. As a particular concern or ours is to develop techniques which can be applied across multiple languages to improve synthetic speech whilst demanding minimal expert supervision, for this year's Challenge we developed a schwa deletion module for use in these 3 languages. Rather than rely heavily on features derived from expert knowledge of any of the individual languages, we detect deletions of inherent vowels during forced alignment and predict them at run time using joint-multigram models [5]. We train our initial forced-alignment models (context-independent Hidden Markov Models of the naively-produced letters described above) by assuming that the inherent vowel is always deleted word-finally and nowhere else. This assumption is then refined by several iterations of forced alignment and model retraining. In the forced alignment phase, the model sequence is updated – acoustic evidence is used to determine whether each instance of an inherent vowel not suppressed by vowel signs or virama is pronounced by the speaker or deleted. The output of the final iteration of forced-alignment is paired with the full letter sequences to provide data for training a joint-multigram model to map from the full letter sequences of words to letter sequences with deleted schwa. This model is used to create annotation of the training data and to produce schwa-deleted letter sequences at run-time.

2.1.3. Unsupervised syllabification

The same simple module used in last year's Challenge for imposing syllable-structure in an unsupervised way was also used to prepare our current entry. As proposed by Mayer [6], we first detected characters corresponding to vowels and consonants with Sukhotin's algorithm [7] from the alphabetised text. The algorithm works on the assumption that in natural languages, vowels and consonants tend to alternate, and the most frequent letter corresponds to a vowel. Then, all word initial consonant clusters with certain minimum frequency were collected to be considered as legal syllable onsets also within words. Syllable boundaries were then placed before the maximal legal onsets within word internal consonant clusters, except that at least one consonant was left as a coda of previous syllable if the cluster contained more than one consonant. Vowel sequences were also split with syllable boundary, if the mutual information of adjacent vowels in text corpus was below a certain threshold.

2.1.4. Word and letter representations

As in our entry for the previous two years, our front-end makes use of no expert-specified categories of letter and word, such as phonetic categories (vowel, nasal, approximant, etc.) and part of speech categories (noun, verb, adjective, etc.). Instead, features that are designed to stand in for such expert knowledge but which are derived fully automatically from the distributional analysis of unannotated text (speech transcriptions and Wikipedia text) are used. The distributional analysis is conducted via vector space models (VSMs); the VSM was originally applied to the characterisation of documents for purposes of Information Retrieval. VSMs are applied to TTS in [8], where models are built at various levels of analysis (letter, word and utterance) from large bodies of unlabelled text. To build these models, co-occurrence statistics are gathered in matrix form to produce high-dimensional representations of the distributional behaviour of e.g. word and letter types in the corpus. Lower-dimensional representations are obtained by approximately factorising the matrix of raw co-occurrence counts by the application of singular value decomposition. This distributional analysis places textual objects in a continuous-valued space, which is then partitioned by decision tree questions during the training of TTS system components such as acoustic models for synthesis or decision trees for pause prediction. For the present voices, a VSM of letters was constructed by producing a matrix of counts of immediate left and right co-occurrences of each letter type, and from this matrix a 5-dimensional space was produced to characterise letters in the alphabetised training text (obtained as described above). Token co-occurrence was counted with the nearest left and right neighbour tokens (excluding whitespace tokens); co-occurrence was counted with the most frequent 250 tokens in the corpus. A 10-dimensional space was produced to characterise word tokens.

Letter representations were used directly as features in decision tree based acoustic modelling. Word representations were used by decision trees (along with other features such as presence of punctuation) to predict pauses at the junctures between words. Data for training these trees are acquired automatically by force-aligning the training data with their transcriptions, and allowing the optional insertion of silence between words.

2.2. Acoustic Modelling

A rich set of contexts was created using the results of the analysis described in section 2.1 for each frame in the training data

for all languages. Features used include the identity of the letter token to which the frame was aligned during forced alignment, and the identities of its neighbours within a 5-letter window. Additional features were the VSM values of each letter in the window, fraction through HMM state, index of state within letter, and the distance from and until a syllable boundary, word boundary, pause, and utterance boundary.

Speech was parameterised with the GlottHMM vocoder [1], consisting of glottal inverse filtering of the speech frame and extracting the 30 line spectral frequency (LSF) coefficients of the vocal tract as well as 10 voice source LSF coefficients, harmonic-to-noise ratio (HNR) of five frequency bands, energy, and fundamental frequency (F_0). All of these features were supplemented with their speed and acceleration coefficients; these were used by a parameter generation algorithm developed for HMM-based speech synthesis [9] to obtain smoothly varying vocoder parameter trajectories from the DNN's output. The resulting synthetic parameter trajectories were post-processed with simple variance expansion [10] and LSF post-filtering. F_0 was processed by interpolating over short unvoiced segments and with wavelet-based enhancement [11], where mainly word-level pitch movement was enhanced, leaving microprosody intact. DNN-based, gender-dependent glottal flow models [12, 13] were taken from last year's Simple4All entry [3].

Data preprocessing and neural network training was performed broadly as for the basic system described in [14]. 95% of the frames labelled as silence were removed. The unvoiced regions of the F_0 track were interpolated, and voicing was represented in a separate stream. Linguistic input features were normalised to the range [0.01, 0.99] and acoustic features standardised.

All systems made use of 6 hidden layers, each consisting of 1024 units with tanh non-linearities, and an output layer with a linear activation function. Network parameters (hidden layer weights/biases, output layer weights/biases) were initialised with small non-zero values, and the network was then optimised with stochastic gradient descent to minimise the mean squared error between its predictions and the known acoustic features of the training set. A small L_2 regularisation was applied to the hidden layer weights. There were 256 frames in each mini-batch. For the first 15 epochs of training, a fixed learning rate of 0.002 was used with a momentum of 0.3. After these initial epochs, the momentum was increased to 0.9 and from that point on the learning rate was halved after each epoch. The learning rate used for the top two layers was half that used for the other layers.

5% of the training utterances in each language were held-out from training for validation purposes; after each epoch, network performance was evaluated on this set. Training finished when performance on the validation set ceased to improve.

2.3. Jointly trained model of duration and vocoder parameters

Our DNN input features make use of the full state alignments obtained during forced alignment with monophone HMMs in that they include the features *fraction of frames through the current state*, and *position of state within the phone*. At synthesis time, therefore, state durations must be determined before full-context annotation can be created. This could be done with an external model, such as a separate neural network or regression tree. We decided instead to experiment with the use of a single model for the prediction of durations and vocoder parameters. Modifications to model training were very simple: we simply appended a 5-dimensional vector to each frame of network outputs, specifying the durations of the 5 states of the phone to which that frame belongs. These 5 extra dimensions of the outputs were mean and variance normalised in the same way as the dimensions corresponding to vocoder parameters, and training proceeded just as if these extra 5 values were vocoder parameters.

Synthesis with a network trained in this way results in a chicken-and-egg problem – to prepare inputs for the network we need state durations, but state durations are only given as outputs of the network. We solve this problem by iteratively refining our state duration predictions. We start with a simple model of state durations: the state duration vector of a phone to be synthesised is predicted as the mean of state duration vectors of all context-independent phones of that type in the training corpus. Using these crude duration predictions, network inputs are prepared and a forward-pass through the network made. All parts of the network's predictions are discarded except for state vector predictions. However, predictions of the state vector of frames within a given phone are typically inconsistent with one another; we therefore take the mean of these prediction within each phone segment and use the mean vector as our final prediction of the state vector, which is used to prepare the final inputs to the network.

3. Results

The identifier for our system in the published results is H.

3.1. Intelligibility

We first consider the results for intelligibility (making use of the published statistical analysis of significant differences between intelligibility of systems at the 1% level with Bonferroni correction). In four languages (Bengali, Hindi, Malayalam and Telugu) there was no system significantly more intelligible than ours. In those 4 languages, there were 4, 2, 3 and 4 systems whose intelligibility was not significantly different from that of our system (specifically, Bengali: B, D, F, J; Hindi: D, F; Malayalam: C, E, J; Telugu: C, E, F, J). Excluding our own system and natural speech, 47 language/team combinations were evaluated for intelligibility (not all possible 48 combinations as no results were published for team I in Marathi). Of those 47, 22 were not significantly more or less intelligible than ours, 22 were significantly less intelligible, and only 3 combinations (all from different teams) were significantly more intelligible.

3.2. Naturalness

We now consider mean opinion scores for naturalness from all listeners on RD sentences; naturalness scores on SUS sentences are not considered here. In Tamil, our system is outperformed only by system F. On three of the languages (Bengali, Marathi and Telugu), our system is outperformed only by two systems. Overall, our system outperforms between 1 and 4 other systems in each language.

3.3. Speaker similarity

Turning to speaker similarity, we now consider mean opinion scores from all listeners on RD sentences; In Tamil and Bengali, our system is outperformed by only one other system. On two of the languages (Malayalm and Telugu), there are only two other systems significantly better than ours. In case of SUS

sentences, our system outperforms every other TTS system significantly in Telugu. In Tamil, there is no other system which is significantly better than ours. Overall, there are between 0 to 3 systems significantly better than ours in any language.

4. Conclusion

Our front-end tools have benefitted from only very limited development since last year's Challenge; the only innovation was the schwa-deletion module, and for the three languages where this was not used, the tools used were identical. In spite of this, our system was among the most intelligible in the challenge, which shows our unsupervised approach is successful in new languages.

The naturalness and speaker similarity of our system have not improved relative to competing systems since last year. This can be attributed to our use of new DNN technology which we have had little experience of tuning for new data sets rather than a well-established highly-tuned HMM system, the use of simpler post-processing without full global variance, and the larger datasets which mean competing concatenative systems can attain better performance.

5. Acknowledgements

This research was supported by EPSRC Programme Grant EP/I031022/1, Natural Speech Technology (NST).

6. References

[1] T. Raitio, A. Suni, J. Yamagishi, H. Pulakka, J. Nurminen, M. Vainio, and P. Alku, "HMM-based speech synthesis utilizing glottal inverse filtering," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 19, no. 1, pp. 153–165, 2011.

[2] O. Watts, A. Stan, Y. Mamiya, A. Suni, J. M. Burgos, and J. M. Montero, "The Simple4All entry to the Blizzard Challenge 2013," in *Proc. Blizzard Challenge 2013*, August 2013.

[3] A. Suni, T. Raitio, D. Gowda, R. Karhila, M. Gibson, and O. Watts, "The Simple4All entry to the Blizzard Challenge 2014," in *Proc. Blizzard Challenge 2014*, September 2014.

[4] 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*.

[5] M. Bisani and H. Ney, "Joint-sequence models for grapheme-to-phoneme conversion," *Speech Communication*, vol. 50, no. 5, pp. 434 – 451, 2008.

[6] T. Mayer, "Toward a totally unsupervised, language-independent method for the syllabification of written texts," in *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, ser. SIGMORPHON '10, 2010.

[7] B. V. Sukhotin, "Optimization algorithms of deciphering as the elements of a linguistic theory," in *Proceedings of the 12th Conference on Computational Linguistics*, ser. COLING '88, 1988.

[8] O. Watts, "Unsupervised learning for text-to-speech synthesis," Ph.D. dissertation, University of Edinburgh, 2012.

[9] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura, "Speech parameter generation algorithms for HMM-based speech synthesis," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP '00. Proceedings. 2000 IEEE International Conference on*, vol. 3, 2000, pp. 1315–1318 vol.3.

[10] H. Silén, E. Helander, J. Nurminen, and M. Gabbouj, "Ways to implement global variance in statistical speech synthesis," in *INTERSPEECH 2012, 13th Annual Conference of the International Speech Communication Association, Portland, Oregon, USA, September 9-13, 2012*, 2012, pp. 1436–1439.

[11] A. Suni, D. Aalto, T. Raitio, P. Alku, and M. Vainio, "Wavelets for intonation modeling in hmm speech synthesis," in *Proc. 8th ISCA Speech Synthesis Workshop*, 2013.

[12] T. Raitio, A. Suni, L. Juvela, M. Vainio, and P. Alku, "Deep neural network based trainable voice source model for synthesis of speech with varying vocal effort," in *Proc. of Interspeech*, Singapore, September 2014, pp. 1969–1973.

[13] T. Raitio, H. Lu, J. Kane, A. Suni, M. Vainio, S. King, and P. Alku, "Voice source modelling using deep neural networks for statistical parametric speech synthesis," in *22nd European Signal Processing Conference (EUSIPCO)*, Lisbon, Portugal, September 2014.

[14] 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 *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2015.