The Simple4All entry to the Blizzard Challenge 2014

A. Suni[1], *T. Raitio*[2], *D. Gowda*[2], *R. Karhila*[2] *M. Gibson*[3], *O. Watts*[4]

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

Antti.Suni@helsinki.fi, tuomo.raitio@aalto.fi, dhananjaya.gowda@aalto.fi, reima.karhila@aalto.fi, Matt.Gibson@nuance.com, owatts@inf.ed.ac.uk

Abstract

We describe the synthetic voices entered into the 2014 Blizzard Challenge by the SIMPLE4ALL consortium. The 2014 Blizzard Challenge presents an opportunity to test and benchmark some of the tools we have been developing 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. Several additions to the system used to build voices for the previous Challenge are described: naive alphabetisation, unsupervised syllabification, and glottal flow pulse prediction using deep neural networks.

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

1. Introduction

We here describe the synthetic voices entered into the 2014 Blizzard Challenge by the SIMPLE4ALL consortium. SIMPLE4ALL is a research project whose goal is to create speech synthesis technology that learns from data with little or no expert supervision.[1] As was true of the previous Challenge's spoke task, this year's Challenge provides an opportunity to test and benchmark technology which has been developed within the project. A central concern of SIMPLE4ALL is how to produce systems in arbitrary new languages with minimal language-specific expertise on the part of the system builders.

A major part of the SIMPLE4ALL software is a set of tools to construct TTS front-ends which make as few implicit assumptions about the target language as possible, and which can be configured with minimal effort and expert knowledge to suit arbitrary new target languages. We consider it important for speech technology to venture beyond the handful of the world's languages where resources such as text normalisers, lexicons and part-of-speech taggers already exist, and our tools support such development. The tools rely on resources which are intended to be universal, such as the Unicode character database, and use unsupervised learning to exploit unlabelled text resources without the need for human annotation. Task IH1 is a test of our tools which tackle this problem, as the consortium members have little language-specific expertise and no language-specific resources for the target languages (Assamese, Gujarati, Hindi, Rajasthani, Tamil and Telugu) beyond the data provided by the Challenge. The tools have been supplemented by two new modules since our previous Blizzard entry. The first of these naively performs alphabetisation of alphasyllabic scripts by making use of the Unicode database, and the second performs unsupervised syllabification from text.

As with our previous Blizzard entry [1], this year's entry makes use of another part of the SIMPLE4ALL toolkit, an implementation of new speech signal models capable of modelling a large variety of speaking styles and vocal emotions [2]. A recent innovation in this part of the toolkit used in the entry is a deep neural network for predicting glottal flow pulse shapes.

2. Hub Task System Description

2.1. Text processing

The tools used for building TTS front-ends for entries to all parts of the challenge are based on the ones used in last year's Challenge [1] with two major extensions.

The only language-specific input to our system is the data used for training. For the IH1 voices, this consisted of the approximately 2 hours of audio data per language plus plain orthography transcription at the utterance level distributed for the Challenge (except for Telugu, in which approximately 3 hours of audio data were made available). In addition, we made use of large quantities of unannotated text data for building word representations in an unsupervised manner – this consisted of approximately 0.7, 3.2, 16.0, 8.6 and 8.4 million tokens of text for Assamese, Gujarati, Hindi, Tamil and Telugu, respectively, which we obtained from Wikipedia. We were not able to obtain any such data for Rajasthani – for this language we built word representations using only the transcripts of the audio data.

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; however, the lightly supervised learning of modules to expand such non-standard words is an active topic of research [3].

For our entry to the previous Challenge, we used a letter-based approach, in which the names of letters are used directly as the names of speech modelling units. In the past this has given good results for languages with transparent alphabetic

[1]www.simple4all.org/

orthographies such as Romanian, Spanish and Finnish [4, 5]. Whilst performance on alphasyllabic Brahmic scripts in last year's challenge was reasonable, we decided to improve intelligibility by using some very general knowledge of script type to create a naive alphabetisation of text input based on characters' names in the Unicode database.

The method used to alphabetise a word is as follows. Take for example the first token occurring in the distributed data for Hindi, *prasid'dha*, meaning '*famous*':

प्रसिद्द

The Unicode representation of this token consists of eight entities 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* actually used for training our system. Note that the same rules were used in all languages, and no language-specific rules were added. For example, no effort was made to predict deletion of schwa in Hindi – the word-final <a> in the above example remained in the alphabetised version of the text, although it is deleted in the spoken form of the word.

A further addition to our system for this year's challenge was a simple module for imposing syllable-structure in an unsupervised way. 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.

Cursory internal evaluation during development showed the alphabetisation described to have a clearly positive effect on resulting synthetic speech. The effect of syllabification seemed to be mixed: a neutral to slightly negative effect on segmental quality is suspected, but possibly accompanied by a slight improvement to prosody.

As in last year's entry, 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 [4], 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.

Note that our system includes two characterisations of letters in different modules – the unsupervised division of letters into vowels and consonants in the syllabification module, and the unsupervised continuous features used to characterise letters in acoustic modelling. There is no reason the vowel/consonant features could not also be used in acoustic modelling directly, but this was not done for the voices submitted to the Challenge.

2.2. Acoustic Modelling

A rich set of contexts was created using the results of the analysis described in section 2.1 for each letter token in the alphabetised training data for all languages. Features used include the identity of the letter and the identities of its neighbours within a 5-letter window. Additional features were the VSM values of each letter in the window, and the distance from and until a word boundary, pause, and utterance boundary.

The GlottHMM vocoder [2] parametrization was used for all voices, consisting of glottal inverse filtering of the speech

Feature	Number of parameters
Energy	1
Fundamental frequency	1
Harmonic-to-noise ratio	5
Voice source spectrum	10
Vocal tract spectrum	30

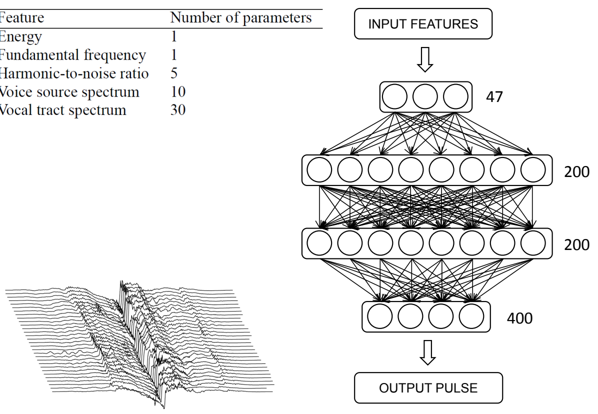

Figure 1: Overview of the DNN-based voice source modelling method used in the hub entries.

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). Similarly to our previous entry, vocal tract LSFs were converted to mel-cepstral representation for HMM training.

For the IH1 voices, the HMM models were trained with the standard HTS 2.0 [8] recipe, modified for additional GlottHMM streams, but using three iterations of decision tree clustering instead of two. For each voice, the voice source estimated by glottal inverse filtering was segmented to two-pitch period glottal flow derivative pulses using glottal closure instant (GCI) detection, windowed with the Hann window, and resampled to a constant length of 25 ms (400 samples at 16 kHz sampling rate). Gender-dependent glottal flow pulse libraries (one for the 5 males and one for the Tamil female) were constructed from the pulses of all six speakers, consisting around 4 million pulses for the male library and one million for the female. Then, a deep neural network (DNN) was trained to create a mapping between the 47-dimensional acoustic feature vectors extracted by the vocoder and the 400-dimensional glottal flow pulses using a method described in [9, 10]. The DNN-based voice source modelling method is illustrated in Figure 1. The DNN consisted a of an input layer with 47 units, two hidden layers each with 200 sigmoid units, and a linear output layer of 400 units, respectively.

At synthesis time, parameter generation is performed considering global variance, with stream-dependent thresholds. Generated mel-cepstral coefficients are converted back to the LSF form for stability checking and vocoding purposes. The generated acoustic features were fed to the trained glottal flow pulse prediction DNN, which generated a glottal flow pulse appropriate for the context. The generated pulses are resampled to length corresponding to the current F_0, scaled in magnitude, and finally noise is added to the pulse in the five frequency bands according to the HNR measure to control the degree of voicing. Finally, the generated pulse sequence is filtered with the vocal tract filter to generate speech.

3. Spoke Task System Description

The spoke task in Blizzard 2014 involves synthesising bilingual speech for a target speaker from a script where native script is interspersed with non-native words, as is commonly encountered in a lot of Indian language texts which contain English words, acronyms and numerals. The main challenge here is that the training data for building a target voice is available only in the native language, and there are no recordings available for the speaker in English.

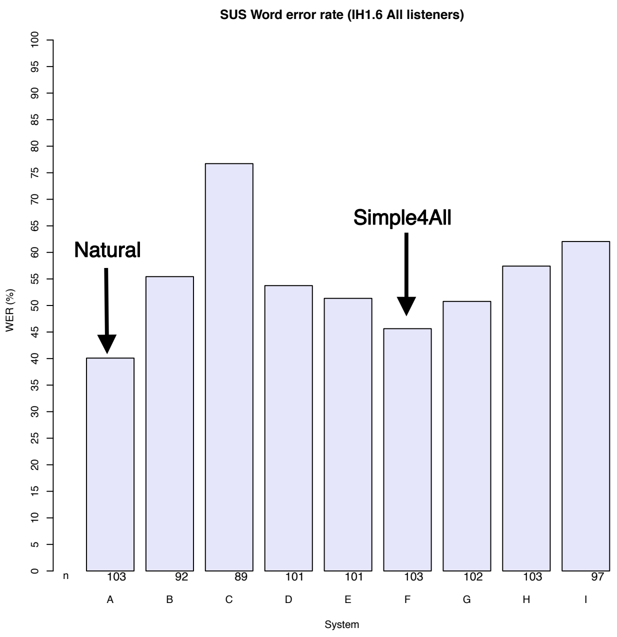

Figure 2: Telugu words error rates on hub task.

3.1. Cross-lingual speaker adaptation with dual frontend

The SIMPLE^{4}ALL system for the spoke task primarily involves adapting an Indian accented English voice to the target speaker in an unsupervised manner. A dual frontend is used for processing the text in Indian languages and English separately. A block diagram of this approach is shown in Fig. 5.

3.1.1. Model Training using Cross-lingual Speaker Adaptation

The systems were built using some of the existing speech synthesis building blocks in a rapid fashion. The English model sets for the speakers were built using the cross-lingual speaker-adaptation tools developed in the EMIME project [11]. An unsupervised approach was used for speaker adaptation, based on probabilistic state mapping [12]. The Indian-accented English-speaker KSP from the Arctic corpus was used as the base voice for adaptation. The voice was trained using full-context labelling provided by the Festival UniGAM module. The model clustering in voice training was done in two steps. In the first step, decision trees were grown with questions referring only to triphone contexts. In the second step, the tree was grown further using the whole set of full-context related questions. Gaussians at the leaf nodes of the tree were trained in the normal way. After the training, a triphone model was generated by combining all the leaves in the decision tree that happened to be under the triphone-related question into a Gaussian mixture model. This model was then used as a phoneme loop recogniser to annotate the Indian language training data with an English phonetic transcription. The triphone transcription was was mapped to a full-context label transcription using a dictionary of alternative pronunciations, and used for training CMLLR adaptation matrices in the standard fashion. To keep the adaptation simple, the deep neural network-based voice source modeling used in the HUB task was not used, and Indian language models were trained separately for this task with the standard GlottHMM features [2].

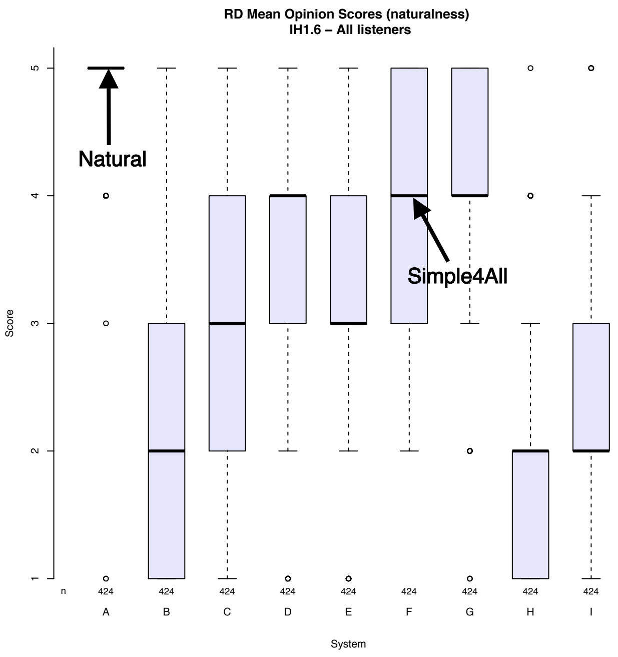

Figure 3: Telugu naturalness scores on hub task.

3.1.2. *Bilingual speech synthesis using dual front-end*

At synthesis time, full-context labels for the bilingual text are generated for the Indian language and English parts of the sentence separately using the dual front-end system. The Ossian front-end is used to process the test sentences wherein the English part is also transcribed in the Indian language script using a pronunciation dictionary and a simple phoneme mapping. Similarly the Festival front-end is used to generate labels for the same sentence now transcribed completely in the Latin script. Now the word count and word boundaries of the two full context label streams are tracked and merged by retaining Indian language labels from the Ossian frontend and the English labels from the festival frontend. The combined model set is then used to synthesize the merged label stream.

4. Results

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

4.1. Hub task

We first discuss the results of our IH1 entries. Based on the scores from all listener types, on no language for which results are made available[2] is there any system significantly more intelligible than ours (published results, 1% significance level), and in each language there is at least one system significantly worse. This is not always the same system: it is variously C, B, H or I, or some combination of those systems. The results plots are too many to show here; we show only the plots of results for Telugu in Figures 2 to 4.

We now consider mean opinion scores for naturalness from all listeners on RD sentences; naturalness scores on SUS sentences are not considered here. On two of the languages (Rajasthani and Tamil), our system is in tied top place with system G. On two languages (Assamese and Telugu), our system is outperformed only by system G. Our system outperforms between 3 and 7 other systems in each language.

[2]All languages except Gujarati, for which results are not available at time of writing

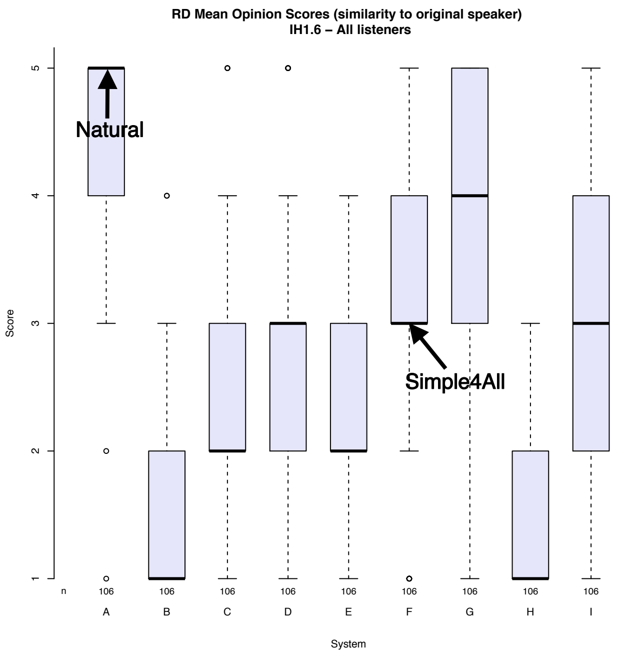

Figure 4: Telugu similarity scores on hub task.

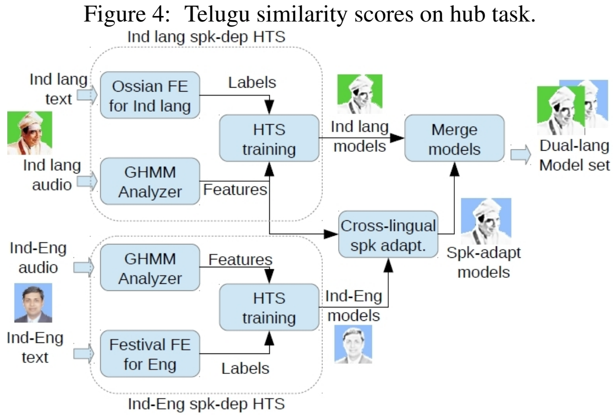

Figure 5: Overview of the SIMPLE4ALL spoke task system.

Turning to speaker similarity, we now consider mean opinion scores from all listeners on RD sentences; naturalness scores on SUS sentences are again not considered here. On two languages (Assamese and Rajasthani) there are no TTS systems significantly better than ours. On the other three languages, there is only one system significantly better than ours. There are between 2 and 5 TTS systems significantly worse than ours in any language.

4.2. Spoke task

Out of the six Indian languages, we tried to complete five languages where in the targe speaker was a male, as we had only one Indian English male voice for the adaptation. Tamil with a female voice was not attempted. Of the five languages attempted, there were problems with the unsupervised speaker adaptation of the Indian English models using Gujarati data. In the case of Assamese, the Assamese models were deemed not good enough due to the pop-noise in the original recordings, which we did not attempt to fix. In the end, submissions for three languages were made, namely, Hindi, Rajasthani and Telugu.

The SIMPLE4ALL system (F) performed on a par with the top systems in terms of naturalness for Hindi and Rajasthani

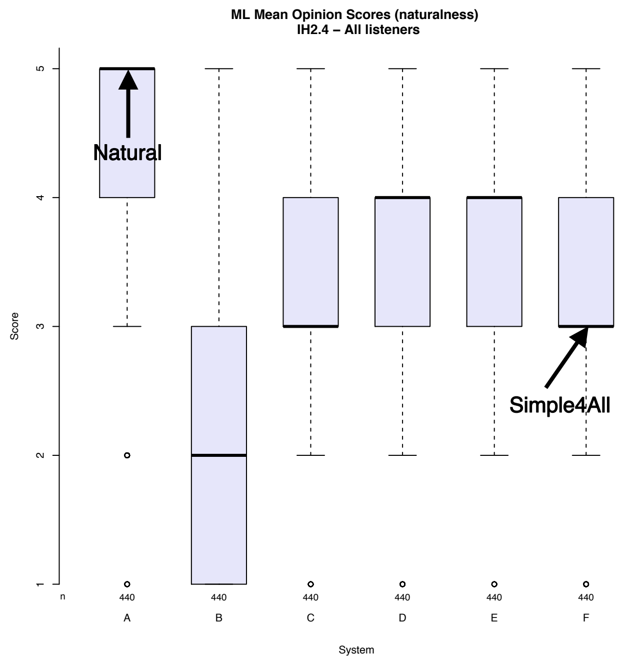

Figure 6: Rajasthani naturalness scores on spoke task.

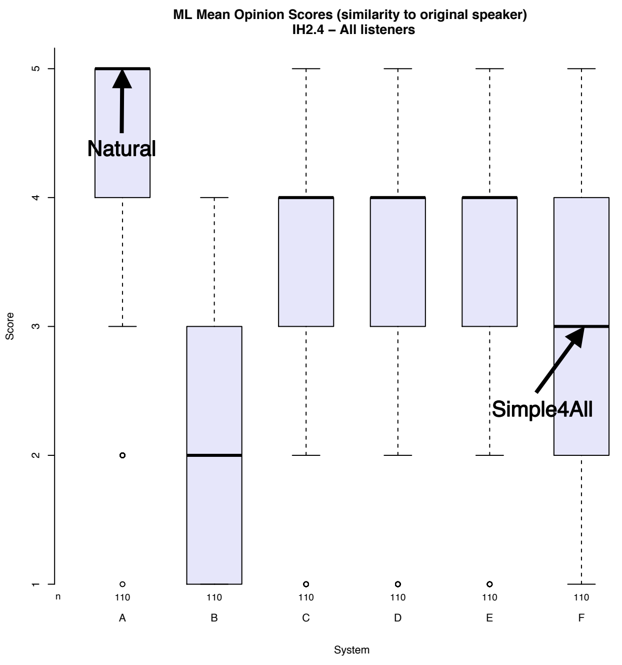

Figure 7: Rajasthani similarity scores on spoke task.

and is second best for Telugu. In terms of speaker similarity, our system (F) performs on a par with the top system in the case of Rajasthani, second best in the case of Telugu, while it is rated last in the case of Hindi. Additional recordings from more Indian-accented English speakers would be required to produce good quality dual-langauage synthesis systems for the rest of the speakers. A plot of spoke task results for our bilingual Rajasthani/English system are shown in Figures 6 and 7.

5. Conclusion

Our system performed well in all parts of the evaluation, supporting the hypothesis that an unsupervised front-end approach is competitive for languages with little infrastructure. At the same time, our Hindi system was not worse than entries from other groups, who we suspect had access to considerable linguistic resources for this language. Our hub task system is as intelligible as any other TTS system across all languages – in no language does any TTS system perform significantly better than ours. We suspect this reflects the impact of the naive alphabetisation used. On two languages in the speaker similarity section of the hub task, no TTS system is significantly better than ours. We consider this – and the respectable performance elsewhere – to be a big success for a TTS system based on vocoding. The current trend to shift from HMM to DNN-based synthesis might yield ongoing performance improvement; in our case, we consider the improvement in speaker similarity is likely due to the DNN-predicted pulses. We consider our success to indicate that statistical parametric synthesis is a good paradigm to follow where only small to medium sized corpora are available.

6. Acknowledgements

This work is supported by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement N$^{\underline{o}}$ 287678.

7. References

[1] 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.

[2] 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.

[3] R. San-Segundo, J. M. Montero, V. Lopez-Ludeña, and S. King, "Detecting acronyms from capital letter sequences in Spanish," in *Proc. Interspeech*, Portland, Oregon, USA, Sep. 2012.

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

[5] A. Black and A. Font Llitjos, "Unit selection without a phoneme set," in *IEEE TTS Workshop 2002*, 2002.

[6] T. Mayer, "Toward a totally unsupervised, language-independent method for the syllabification of written texts."

[7] B. V. Sukhotin, "Optimization algorithms of deciphering as the elements of a linguistic theory."

[8] H. Zen, T. Nose, J. Yamagishi, S. Sako, T. Masuko, A. Black, and K. Tokuda, "The HMM-based speech synthesis system (HTS) version 2.0," in *Proc. of 7th ISCA Workshop on Speech Synthesis*, 2007, pp. 294–299.

[9] 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, accepted.

[10] 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, accepted.

[11] M. Kurimo, W. Byrne, J. Dines, P. N. Garner, M. Gibson, Y. Guan, T. Hirsimäki, R. Karhila, S. King, H. Liang, K. Oura, L. Saheer, M. Shannon, S. Shiota, J. Tian, K. Tokuda, M. Wester, Y.-J. Wu, and J. Yamagishi, "Personalising speech-to-speech translation in the EMIME project," in *Proc. of the ACL 2010 System Demonstrations*, Uppsala, Sweden, July 2010.

[12] M. Gibson, T. Hirsimäki, R. Karhila, M. Kurimo, and W. Byrne, "Unsupervised cross-lingual speaker adaptation for hmm-based speech synthesis using two-pass decision tree construction," in *ICASSP*, March 2010, pp. 4642–4645.