THE CEREVOICE SPEECH SYNTHESISER

Juan María Garrido[1], Eva Bofias[1], Yesika Laplaza[1], Montserrat Marquina[1]
Matthew Aylett[2], Chris Pidcock[2]

[1]Barcelona Media Centre d'Innovació, Barcelona, Spain

[2]Cereproc Ltd, Edinburgh, Great Britain

ABSTRACT

This paper describes the CereVoice® text-to-speech system developed by Cereproc Ltd, and its use for the generation of the test sentences for the Albayzin 2008 TTS evaluation. Also, the building procedure of a Cerevoice-compatible voice for the Albayzin 2008 evaluation using the provided database and the Cerevoice VCK, a Cereproc tool for fast and fully automated creation of voices, is described.

1. INTRODUCTION

CereVoice® is a unit selection speech synthesis software development kit (SDK) produced by CereProc Ltd., a company based in Edinburgh and founded in late 2005 with a focus on creating characterful synthesis and massively increasing the efficiency of unit selection voice creation [1, 2, 3, 4, 5].

Cereproc Ltd and *Barcelona Media Centre d'Innovació* (BM) started in 2006 a collaboration which led to the development of two text normalization modules, for Spanish and Catalan, the lexicon and the letter-to-sound rules for transcription in both languages, and a Spanish-Catalan bilingual voice ('mar') for the CereVoice® system. As a result of this collaboration, BM became an official 'voice developer' of Catalan and Spanish voices for CereVoice®, and got an academic license to use the CereVoice® system for research purposes.

In this paper a brief description of the CereVoice® TTS-system engine is given, and the procedures of voice building and speech files generation for the Albayzin 2008 evaluation, carried out by BM with the support of Cereproc, are described.

2. GENERATING SPEECH USING CEREVOICE

To generate speech using the CereVoice® system, three components are necessary: the Cerevoice engine (Cerevoice SDK), a text normalization module (the one provided by Cereproc or any other compatible with Cerevoice), and a TTS Voice. Also, some optional modules, such as user lexicons or user abbreviations tables, can be used to improve the text processing in particular applications. Figure 1 shows a workflow scheme of the system.

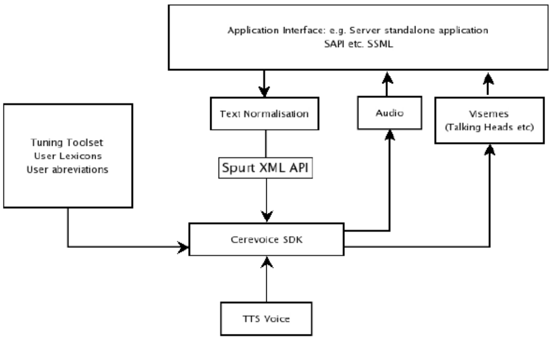

Figure 1. *Overview of the architecture of the Cerevoice synthesis system. A key element in the architecture is the separation of text normalization from the selection part of the system and the use of an XML API.*

In the following subsections, a brief description of the main features of the Cerevoice engine, the text processing module and the voices is given.

2.1. Cerevoice engine

Cerevoice is a new faster-than-realtime speech synthesis engine, available for academic and commercial use. The system is designed with an open architecture, has a footprint of approximately 70Mb for a 16Khz voice and runs at approximately 10 channels realtime. The core Cerevoice engine is an enhanced synthesis 'back end', written in C for portability to a variety of platforms. To simplify the creation of applications based on Cerevoice, the core engine is wrapped in higher level languages such as Python using Swig.

The Cerevoice core engine is a diphone based unit selection system with pre-pruning and a Viterbi search for selecting candidates from the database. The system uses both symbolic (e.g. stress, break index information) and parametric target cost functions (e.g. F0 and duration). Transition costs are based on Line Spectral

Frequencies, F0 smoothed over voiced plus unvoiced speech, and energy. Target and transition weights can be set manually for fine tuning of each particular voice.

The engine does not fit the classical definition of a synthesis back end, as it includes lexicon lookup and letter-to-sound rule modules. The Spanish and Catalan versions of these lexica and rules were jointly developed by BM and Cereproc.

An XML API defines the input to the engine. The API is based on the principle of a 'spurt' of speech. A spurt is defined as a portion of speech between two pauses. The XML API defines also the set of tags that the engine is able to interpret to control several speech output parameters, such as tempo, global tone, genre, or even the language.

One of the aspects that can be controlled through the insertion of tags in the input text, and that can be used to improve the quality of the output speech, is the use of *variants*. In Cerevoice it is possible to ask the engine to prune out a section of the best path found during the Viterbi search and to rerun the Viterbi over that section to find a less optimal alternative or variant. Inside an XML spurt, a word can enclosed by a 'usel' tag containing a variant attribute to force this behaviour. For example <usel variant='0'> is equivalent to no tag, and <usel variant='6'> would be the sixth alternative according to the Viterbi search.

2.2. Text normalization module

The Cerevoice engine is agnostic about the 'front end' used to generate spurt XML. However, the Cerevoice system includes its own modular Python system for text normalization in several languages. Those modules include a set of normalization rules for processing hours, dates, telephone numbers, figures, abbreviations, letters and any other element that needs to be expanded.

The Spanish and Catalan modules provided within the Cerevoice system where jointly developed by Cereproc Ltd and BM. The Spanish text processing module has been used for the generation of the test sentences for the evaluation.

2.3. Cereproc Voices

TTS voices for the Cerevoice SDK are currently available in English (Scottish and British), Catalan, Spanish and Japanese. The Spanish and Catalan voices are the result of a collaboration between BM and Cereproc Ltd.

Voice building is carried out using the Cerevoice Voice Creation Kit (VCK), a tool designed and tuned for fast development of new voices. Using this tool, voice building is a heavily automated, modular, dependency-driven process, consisting of two main types of component: speech parameterization and segmentation. Speech data, text transcriptions, and a lexicon are the only required inputs to voice building.

Voice building includes often a second stage of voice tuning. Voice tuning consists of a manual adjustment of the weights for the target and transition cost functions used for unit selection during the synthesis process. This process involves iterative trial-and error modification of the weights in order to set the optimal combination.

3. THE CEREVOICE VOICE CREATION KIT

The Cerevoice VCK is a tool for fast voice building with minimal manual intervention of the voice developer. Only the speech data, in RIFF wav files, the corresponding orthographic transcription of each utterance, as a single UTF-8 text file, and a lexicon of the language are needed as input. It runs on Linux machines and requires Python for running.

CereProc recommends 15 hours of data for a general purpose voice, although around 5-6 hours of audio data should be enough for an acceptable voice. Voices can also be built incrementally, adding data gradually, until the required quality is achieved.

Cerevoice VCK expects the orthographic text to be provided in the form of a recording script, a text file containing a text line for each wav file, and an identifier preceding the text. If CereProc's Voice Recorder software is used for the recordings, a valid recording script is also needed to display the texts to be read by the speaker, and to name and store automatically the corresponding wav files. If speech data coming from an external source are used for voice building, orthographic transcriptions have to be merged into a single script, in the format required by the VCK.

Examples of valid script lines are:

```
v0001_001 Acme Limited is the best company
in the wold.
v0001_002 Acme Limited (company code A C
M) made one point five billion dollars
profit last year.
```

Finally, a valid lexicon is also needed for the building process. CereProc provides a lexicon for every supported languages and accents. If additional sentences have been added to the VCK recording script, it may be necessary to add words to the pronunciation lexicon. During voice building, sentences are excluded from the voice if a word does not exist in the lexicon.

The building process needs also some additional information about the location of the script and speech files, and other related information, which has to be set in an XML configuration file. The creation of such a configuration file is also one of the tasks needed to build a new voice.

Voice building consists of two steps: segmentation and parameterization. In the first one, segmentation is carried out using the HTK Hidden Markov Model toolkit in forced alignment mode. In the second one, F0 and pitch mark parameters are generated using the ESPS tools ’epochs’ and ’get f0’. Edinburgh Speech Tools’ ’sig2fv’ is used to generate cepstral parameters, which are used to generate Line Spectral Frequencies.

During the segmentation of the speech, bad data may be thrown out of the build. This may happen if there are noisy or truncated audio files, if the speech does not match the text in the script, files where the lexicon pronunciation for a word is incorrect, or for files where the speaker has mispronounced a word. VCK provides several ways of recovering and inspecting the discarded files, in order to fix input problems.

A speech GUI is also provided to allow the developer to find data errors such as lexicon problems and mismatches between audio files and text while running the voice.

4. BUILDING THE ALBAYZIN EVALUATION VOICE

The Albayzin 2008 TTS evaluation has been considered at BM, first of all, as an excellent chance to compare Cerevoice with other systems. But it has been seen also as a way to test the capabilities of the Cerevoice VCK to build voices from external data in a fast way, and with a minimum of manipulation of the input data. And finally, considering that the database used for this evaluation contains only two hours of speech, the evaluation has been used to test the behavior of the Cerevoice engine with voices build from a small amount of speech data (the usual amount of speech used to build commercial voices is about 5-6 hours).

According to these considerations, the building procedure of the voice involved the usual steps when creating a voice with the Cerevoice VCK from external data:

a. preparation of the script file;
b. preparation of the wav files (renaming the files to have a valid name);
c. preparation of the configuration file;
d. running of the VCK;
e. checking and fixing of errors.

Creating the script file was a straightforward task, which involved deletion of the pause marks in the files, merging all the text files into a single file, adding a VCK-compatible identifier to each line, and translating the resulting file to UTF-8 format.

Preparing the wav files was also fast, and involved renaming the files with the corresponding VCK-compatible identifier (the same as the corresponding orthographic text line in the script file), and preprocessing of the file to perform a peak normalization of the files. This task was carried out using an audio processing tool provided by Cereproc as part of the VCK. We decided to include the whole set of sentences, although they came from three different 'styles' (paragraphs, sentences and literary) due to the small global size of the database. However, we decided to identify them as different genres, Questions were not considered a different genre, as it was the case in the ‘mar’ voice building procedure, because it was not possible to take advantage of this differentiation for the generation of the sentences (it was not allowed to use genre tags).

Finally, a configuration file was also prepared. To do this, a signature file was provided by Cereproc. This is a necessary step for the creation of any new voice.

Once all the necessary material was ready for processing, VCK was launched for voice building. This process was really fast: about half an hour in an Intel-based server. No special problems arose during the building procedure, so it was not necessary to rebuild the voice several times.

No special tuning of the weights was carried out for the building of the voice. The ones used are those established for the building of the ‘mar’ Spanish voice.

5. GENERATING THE TEST SENTENCES

Test sentences were generated using the latest available version of the Cerevoice SDK (2.1.0). The text normalization module developed by Cereproc and BM for Spanish was used for text processing. No modification of the input text files was carried out during the generation process, as established in the evaluation requirements, apart from their conversion to valid XML files in UTF-8, to allow Cerevoice to process them. This condition excluded the insertion of variant tags in the text files.

6. CONCLUSIONS

In this paper, the CereVoice® TTS system and the Cerevoice Voice Creation Kit have been presented. Also, the building procedure of the evaluation voice and the generation of the test sentences have been described. The building procedure has shown the capabilities of the VCK to create voices for CereVoice® with minimal effort and manual intervention. The results of the evaluation will show to what extent the CereVoice® system is able to generate high-quality synthetic speech when the amount of data available for voice building is smaller than usual.

7. REFERENCES

[1] M.P. Aylett and Yamagishi, J., "Combining Statistical Parametric Speech Synthesis and Unit-Selection for Automatic Voice Cloning". LangTech 2008, Rome.

[2] M.P. Aylett and C.P. Pidcock, “The CereVoice Characterful Speech Synthesiser SDK (Industrial Demo)”. IVA 2007, Paris, France, Proceedings. Lecture Notes in Computer Science 4722 Springer.

[3] M.P. Aylett, J.S. Andersson, L. Badino and C.J. Pidcock, “The Cerevoice Blizzard Entry 2007: Are Small Database Errors Worse than Compression Artifacts?”, Blizzard Challenge Workshop, Bonn, 2007.

[4] M.P. Aylett and C.P. Pidcock, “The CereVoice Characterful Speech Synthesiser SDK”, AISB 2007, Newcastle. pp.174-8

[5] M.P. Aylett, C.P. Pidcock and M.E. Fraser, “The CereVoice Blizzard Entry 2006: A prototype Database Unit Selection Engine”, Blizzard Challenge Workshop, Pittsburgh, 2006.