Embedded Cantonese TTS for Multi-Device Access to Web Content

Tien-Ying Fung, Yuk-Chi Li*, Eddie Sio*, Icarus Lee*, Helen Meng*, P. C. Ching***

*Human-Computer Communications Laboratory,
Department of Systems Engineering and Engineering Management,
**Digital Signal Processing & Speech Technology Laboratory,
Department of Electronic Engineering,
The Chinese University of Hong Kong,
Hong Kong SAR, China

{tyfung, ycli, ktsio, hylee, hmmeng}@se.cuhk.edu.hk, pcching@ee.cuhk.edu.hk

Abstract

This paper describes the development of an embedded Cantonese text-to-speech synthesizer to enable multi-device access to Chinese Web content. Advancements in wireless communication is driving Web visitors from using desktop PCs to mobile handheld devices. Significant reduction in the form factors of the client devices tends to shift information delivery from the visual to the aural modality. This calls for synthesizers that can run on relatively stringent computational and storage resources of handheld devices. We report on the migration of our Cantonese synthesizer, CU VOCAL, from the desktop to the embedded platform. Migration preserves the support for speech synthesis markups (SSML), ensures code compatibility and lowers the storage requirements of the syllable inventory. Results from listening tests indicate no signification deterioration in synthesis quality of embedded CU VOCAL when compared to its desktop counterpart.

1. Introduction

A recent effort in our group aims to enable multi-device accessibility to Chinese Web content through the development of the Author Once, Present Anywhere (AOPA, www.se.cuhk.edu.hk/~aopa) software platform. AOPA is illustrated in Figure 1. We adopt the W3C standards in using Extensible Markup Language (XML) for encoding the unified content repository and Extensible Stylesheet Language family (XSL) for specifying the presentation style(s) of the content to suit client devices of a variety of form factors. When the form factor of the client device is significantly reduced, information delivery shifts from the visual modality to the aural modality. This motivates our work in developing an embedded text-to-speech synthesizer in support of multimodal information delivery on mobile handheld devices.

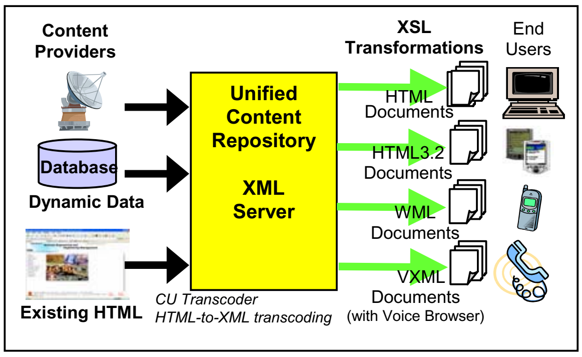

Figure 1. The AOPA software platform aims to support multi-device accessibility to Web content.

The basis of our development is the CU VOCAL Cantonese text-to-speech synthesis engine. Cantonese is a major dialect of Chinese, spoken by over 60 million people in Hong Kong, Macau, South China and many other Chinese communities. Its rich tonal structure presents interesting research issues in TTS. CU VOCAL [1-4] is a synthesis engine that adopts a syllable-based concatenative approach to generate highly intelligible and natural synthetic speech from free-form Chinese text input. CU VOCAL has previously been integrated into the AOPA platform a server-side (or desktop PC) utility and a Web Service [4] in order to support information delivery to Pocket PCs or smart phones. However, the performance of the CU VOCAL Web Service [5] is highly affected by the speed and stability of the network. There is a dire need to develop embedded synthesizers [6] that can run locally on the handheld device. As we migrate from the server/desktop platform towards an embedded platform, we need to develop the engine to run on stringent computational and storage resources without significantly compromising the quality of the synthesized speech. In the following, we elaborate on the migration process, which preserves the support for speech synthesis markups (SSML), ensures code compatibility and lowers the storage requirements of the syllable inventory.

2. Desktop Version – Support for W3C SSML

We have recently developed CU VOCAL to support W3C's Speech Synthesis Markup Language (SSML) version 1.0 [7]. SSML aims to provide a standardized way for authoring content for speech synthesis, with special provisions that allow authors to specify aspects of synthesis such as pronunciation, volume, pitch, etc. There are a total of seventeen elements in SSML 1.0. Of these, CU VOCAL supports nine elements [8] that are directly related to text-to-speech synthesis (see Table 1).

Categories	*SSML elements*
Document Structure, Text Processing and Pronunciation	**p** (paragraph) and **s** (sentence), **say-as, phoneme, sub,** speak, meta, metadata, xml:base, xml:lang, lexicon
Prosody and Style	**prosody, emphasis, voice, break**
Other Elements	audio, mark, desc

Table 1. Elements in W3C's SSML version 1.0 [7] and their categorizations. Those supported by CU VOCAL in boldface. More specifically, the phoneme element can be used to specify the pronunciation of a Chinese character with Cantonese Jyutping.[1] The prosody element can be used to adjust the pitch, rate and volume of synthetic speech.

[1] Jyutping (粵拼) is a Cantonese pronunciation transcription scheme developed by the Linguistic Society of Hong Kong.

In addition to tags shown in Table 1, we have designed several extra tags to cater for Cantonese Chinese synthesis, such as `tone` to indicate a tone change for a Chinese syllable, `phrase` and `word` to define the phrase and word boundaries.

3. Migration from Desktop PC to PDA

CU VOCAL was previously designed to run on a desktop PC Windows platform. The current work aims to engineer CU VOCAL to run on a PDA platform in order to support multi-device and mobile access to Chinese Web content. The desktop version of CU VOCAL has about 15MB runtime memory consumption and around 80MB storage for the synthesis (syllable) inventory and lexicon. Our development platform for PDA (Pocket PC Windows CE) has a PXA scale CPU with about 64 MB storage. Hence we need to seek methods to reduce the storage requirements of CU VOCAL by at least two-fold.

The desktop version of CU VOCAL consists of several components to support the following process control flow: Input Chinese text is parsed and interpreted for any SSML tags. This is followed by pronunciation lexicon lookup, where the lexicon is loaded into memory during system initialization. The pronunciations are represented in the form of Cantonese syllables and drive the unit selection component to retrieve the most fitting syllable unit from the inventory. The selected syllables are then concatenated together to generate the output speech. During the synthesis process, signal modifications, such as duration lengthening, may be incorporated to the synthesized speech as specified in the SSML tags. This process control flow is depicted in Figure 2.

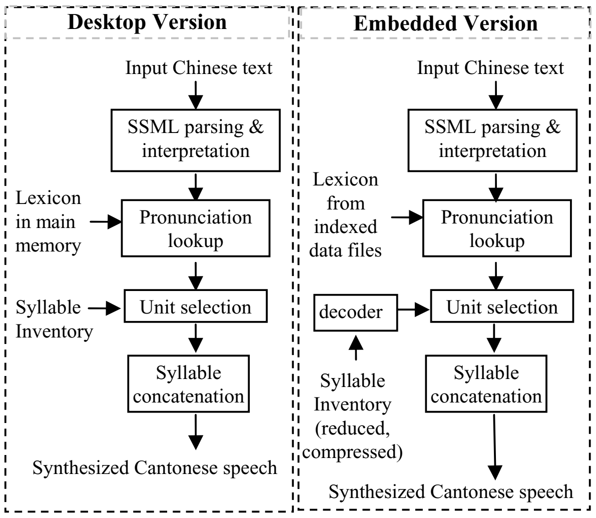

Figure 2. Architectures of the desktop and embedded versions of CU VOCAL. The desktop version loads the lexicon into main memory, while the embedded version accesses the lexicon via file I/O. Also, the embedded version includes a decoder to operate with a compressed syllable inventory.

The embedded version of CU VOCAL preserves the same functionalities as the desktop version. Development effort is devoted to (1) code compatibility for running on an embedded platform, (2) reduction and compression of the syllable inventory. These three aspects will be elaborated in the following sections.

4. Code Compatibility for Embedded Platform

CU VOCAL continues to support SSML in its embedded version. This is achieved by migrating the SSML parser to the embedded environment based on the Expat parser [9], an open source XML parser.

Pronunciation lookup in the desktop version of CU VOCAL consults a lexicon of over eighty thousand entries. However, on an embedded platform it becomes too costly to load such a lexicon into main memory as the available storage is constrained and the loading time will likely be too long (on the order of *several* minutes). An alternate implementation is needed for fast pronunciation retrieval. Hence, we adopted the inverted file technique, commonly used in information retrieval [10], to index the lexicon. A hashing function is used to generate a hash key for each lexicon entry. The hash key is used to find the location of entries in the indexed file. Lexicon entries sharing the same hash key are stored sequentially in the indexed file. This technique allows direct file access for pronunciation lookup and shortens system initialization time to fewer than two seconds.

5. Reduction and Compression of the Syllable Inventory

A major need in migrating from the desktop to the embedded platform is to significantly lower the storage size required by the syllable inventory. We wish to preserve the quality of the recorded speech at 16kHz sampling rate and 16-bit PCM. Hence we use the strategies of (i) reducing the number of tokens stored for every syllable and (ii) compressing the syllables in storage. The procedures are as follows:

5.1 Reducing the number of Syllable Tokens

Multiple tokens are stored in the original CU VOCAL syllable inventory to cover contextual variations due to positional, tonal and coarticulary features [1-4]. There are five positional features for capturing the declination effect – START and END cover first and final syllable positions respectively, and the positions in between them are covered by NEAR-START, CENTER, and NEAR-END in roughly equal portions. The tonal features cover the six non-entering tones among the Cantonese nine-tone system (see Figure 3). This is because entering tones primarily vary the duration but not the tone height of a syllable. The coarticulatory features are mainly due to place of articulation. Given that there are approximately 1700 tonal syllables in Cantonese, our reduction strategy aims to store no more than *four* tokens for every syllable in order to fit the embedded platform. If the original inventory has four or fewer tokens, then all of them are transferred into the reduced inventory. Otherwise we apply a *selection scheme* that is designed to minimize excursions in the tone trajectory of a synthetic utterance as well as maintain relative syllable durations according to positional effects. We consider these factors to be most important for the perception of tonal languages such as Cantonese and decided to tradeoff the positional and tonal features with the coarticulary features. This selection scheme is realized in terms of five main rules:

Rule 1: Select the syllable token in the START position

A syllable token realized in the START position (i.e. sentence-initial position, denoted by START-syl) tend to have higher f0 levels. They can also preserve their tone shape better due to the lack of left tonal context. These tokens are desirable for concatenative synthesis at sentence-initial positions. Should multiple START-syl tokens exist in the original inventory due to different right contexts, one of the tokens will be selected at random for the reduced inventory. This is because previous investigation involving listening tests shows that mismatches in right tonal context do not have significant effects in concatenative synthesis [3]. Hence the current rule (Rule 1) may select up to one token to fill the four available slots assigned to a given syllable. Applications of the subsequent rules will fill up the remaining available slots.

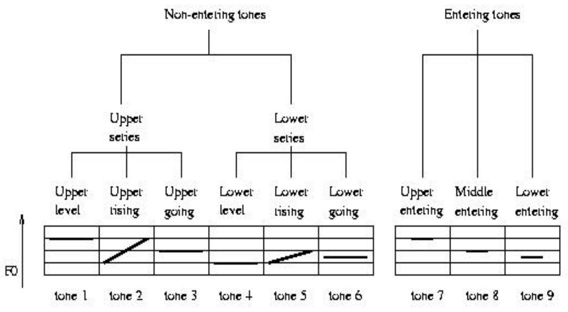

Figure 3. The Cantonese nine-tone system.

Rule 2: Select the syllable token in the END position

A syllable token realized in the END position (i.e. sentence-final position, denoted by syl- END) tend to have lower f0 levels due to the declination effect and longer durations due to pre-pausal lengthening. If only a single token for this particular END-syl is present in the original inventory, it is selected for the reduced inventory. However, it is likely that multiple tokens for this particular syl-END are available due to different left contexts. Under such circumstances, we favor left contexts with neutrality in tone heights as well as level tones. Hence the selection rule based on the left tonal context follows the order of preference:

Tone 3 > Tone 6 > Tone 1 > Tone 4 > Tone 5 > Tone 2

The current rule (Rule 2) may add no more than one token to the available slots in the reduced inventory. Remaining slots are filled by applying subsequent rules.

Rule 3: Select the syllable token with minimum excursion in the tone trajectory due to the left tonal context

This rule is consistent with the previous rule in favoring left contexts with neutrality in tone heights and level tones. Hence Tone 3 in the left context is selected with high priority and the token is denoted by (Tone 3)-syl. If several such tokens are available in the original inventory, we favor the positional feature CENTER over NEAR-START or NEAR-END. Hence the order of preference is:

(Tone 3, CENTER)-syl > (Tone 3, NEAR-START or NEAR-END)-syl

If the original inventory does not contain such tokens, we revert to the next preference as described in Rule 2, i.e.:

(Tone 6, CENTER)-syl > (Tone 6, NEAR-START or NEAR-END)-syl

The current rule (Rule 3) may add no more than one token to the available slots in the reduced inventory. Remaining slots are filled by applying subsequent rules.

Rule 4: Select the syllable token that complements its selected counterparts in Rules 1 to 3

This rule selects for one of the remaining slots in the four allowable slots for a given syllable. The selected token aims to complement its counterparts that have already been selected by the previous three rules. Hence tokens of the form START-syl, syl-END, (Tone 3)-syl and (Tone 6)-syl are not considered as possible additions.

If (Tone 3)-syl has previously been selected to occupy one of the slots, then the current rule will select (Tone 4)-syl provided that a token with such left context exists in the original inventory. This is because Tone 3 belongs to the "upper" series in non-entering tones (see Figure 3) and we need to use Tone 4 to cover the "lower series" as well. As such both the upper and lower tone heights are covered.

Likewise, if (Tone 6)-syl has been selected previously, the current rule will select (Tone 1)-syl. This is because Tone 6 belongs to the "lower" series in non-entering tones and we need to use Tone 1 to cover the "upper series" as well. As such both the upper and lower tone heights are covered.

If both tokens, i.e. (Tone 4)-syl and (Tone 1)-syl are available in the original inventory and if there are at least two unfilled slots in reduced inventory, the current rule will select *both* tokens for the reduced inventory.

Rule 5: Select the syllable token whose left context has a rising tone trajectory

Having applied Rules 1 to 4, should there still be open slots for a given syllable, the current rule will select a token with rising tone trajectory in its left context.

If the token (Tone 3)-syl has not been selected previously, the current rule will select the token (Tone 5)-syl. This is again due to the lesser degree of excursion in Tone 5 when compared to Tone 2. Furthermore, Tones 3 and 5 end with similar heights and tend to have similar effects as left context. At this point, if there still remains unfilled slot(s) in the reduced inventory, the current rule will continue to select (Tone 2)-syl as well. This is because Tone 2 has the maximum excursion in tone trajectory among all the tones.

If (Tone 3)-syl has previously been selected, the current rule will select (Tone 2)-syl directly.

These five rules in our selection strategy reduce the syllable inventory of CU VOCAL from 80MB to 40MB.

5.2 Compressing the Reduced Syllable Inventory

This section describes the compression process that further lowers the storage size of CU VOCAL's syllable inventory from 40MB to 27MB.

We applied the data compression facility from Ogg Vorbis [11].[2] It meets our requirements of high compression rate, fast compression and decompression times while sustaining a reasonably high sound quality after compression and decompression.

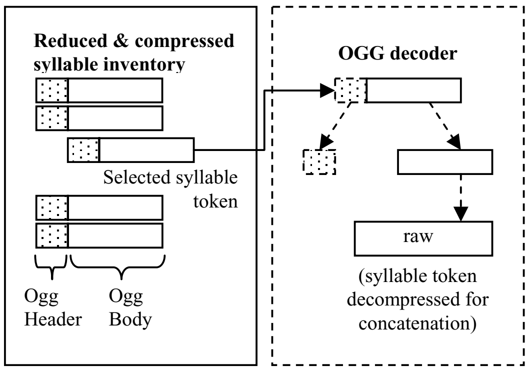
Reduced & compressed syllable inventory
Selected syllable token
Ogg Header
Ogg Body
OGG decoder
raw
(syllable token decompressed for concatenation)

Figure 4. Ogg Vorbis compression/decompression in embedded CU VOCAL.

The encoder and decoder of Ogg Vorbis are available for download [12] as free and open source software for the desktop platform. We engineered the Ogg decoder for the embedded platform. The syllable tokens in the reduced syllable inventory are originally in raw format. They are encoded (compressed) into Ogg Vorbis format and each is given a header (see left inset in Figure 4). During concatenative synthesis, when a syllable token/unit is selected, the decoder removes the header and decompresses the body to restore the unit into its original raw format, which is then used for concatenation (see right inset of Figure 4). The reduced and compressed syllable

[2] We have also experimented with MP3 [12] compression and decompression but found that it appends variable segments of silence to the end points of our syllable tokens which is undesirable for concatenative synthesis.

inventory and the Ogg Vorbis decoder are integrated with the other modules in CU VOCAL to give the new embedded version of the synthesizer.

The reduction strategy and data compression methods described in this section brings the syllable inventory of CU VOCAL from 80MB to 27MB which is an overall reduction of approximately 66%.

6. Listening Tests

To evaluate the performance of embedded CU VOCAL, we conducted two listening tests. The first test assesses the effectiveness of our selection scheme designed for syllable inventory reduction. The second evaluates the comparative speech qualities between the desktop and embedded versions of CU VOCAL.

6.1. Evaluating the Selection Scheme designed for Syllable Inventory Reduction

We generated two reduced syllable inventories from the original CU VOCAL inventory. The first one is generated with the selection scheme described in section 5.1, while the other involves random selection of up to four tokens for every syllable. Ten sentences were selected from a news article and synthesized with the two reduced syllable inventories. This produces two audio clips for every sentence. Ten subjects were invited to evaluate the naturalness of ten pairs of audio clips on a 7-point Likert Scale: 1(barely natural) to 7 (very natural). Each pair of audio clips was played to the subjects twice in randomized order. For the reduced inventory generated with random selection, the mean opinion score (MOS) is 3.9. For the reduced inventory generated with our selection strategy, the MOS is 4.5. The difference was statistically significant (α=0.05) and indicates the effectiveness of our selection strategy.

6.2. Listening Comprehension and Quality of Synthesis

In order to evaluate the intelligibility of the output from the embedded CU VOCAL, we used it to transform an entire news article into synthesized speech, which is then used in a listening comprehension test. Twelve subjects were invited to joint he test. The duration of the synthesized speech was approximately one minute and was played twice for the subjects, who were also free to take notes. After listening, the subjects were presented with five questions that asked for specific information in the news article. The article and the questions are shown in Tables 2 and 3 respectively. The subjects were asked to write down their answers for scoring – 0 point for errors, 1 point for a partially correct answer and 2 points for a correct answer. The average score is 9.3 out of 10 points.

本港環保團體綠色和平進行 一項研究發現，新界粉嶺電子垃圾處理場的土壤樣本含鉛量，超出國際標準 5 至 10 倍。該團體發言人擔心，全港 91 個電子垃圾處理場同樣面對有毒物質滲漏問題，恐進入食物鏈後，最終毒害港人，建議政府盡快收緊法例，阻此輸入外國電子垃圾，以及禁止香港繼續成爲電子垃圾轉口地，以防污染香港與內地環境。

Table 2. News article for the listening comprehension test.

We also ran a contrastive experiment that compares the desktop and embedded versions of CU VOCAL. Both are used to generate synthesized speech that reads out the same news article. Another fifteen subjects were asked to rate the naturalness of the two synthesized outputs based on a 7-point Likert scale: 1 (barely natural) to 7 (highly natural). The MOS for embedded CU VOCAL was 4.5, while that for the desktop CU VOCAL was 4.9. These results suggest that there is no significant deterioration as we migrate from the desktop CU VOCAL to the embedded version.

1. 報導指出哪個本港團體進行了一項研究？請寫出團體名稱·
2. 研究報告提及位於新界哪裡的電子垃圾處理場？
3. 那裡土壤的含鉛量超出國際標準多少倍？
4. 該團體擔心全港其他電子垃圾處理場會同樣面對甚麼問題？
5. 他們建議政府收緊法例以及禁止香港繼續成爲甚麼？

Table 3. Questions in the listening comprehension test.

7. Summary and Conclusions

This paper reports on our recent effort in enabling multi-device access to Chinese Web content. Such content needs to be accessible not only via desktop PCs, but also handheld PDAs or smart phones with much reduced screen sizes. When the client device has a small form factor, information delivery often migrates from the visual modality towards the aural modality. This calls for text-to-speech synthesis technology. PDAs typically have tighter constraints in terms of computing power and storage space when compared with desktop PCs. This motivates our work in the migration of our Cantonese text-to-speech synthesizer, CU VOCAL, from the desktop to the embedded platform. Migration ensures code compatibility and uses the inverted file technique to index the lexicon for fast system initialization and rapid pronunciation lookup. Major effort is devoted towards reducing the size of the syllable inventory that is used for concatenative synthesis – this is achieved by designing a syllable selection strategy that extracts no more than four tokens for every syllable from the original syllable inventory into the reduced inventory. The reduced syllable inventory is further compressed by the Ogg Vorbis facility. Syllable inventory reduction and compression lowered the storage requirement of CU VOCAL from 80MB to 27B in the desktop and embedded versions respectively, i.e. a 66% reduction. Listening tests indicate that the embedded version suffers no significant deterioration in synthesis quality when compared with its desktop counterpart.

8. Acknowledgments

This work is substantially supported by the Innovation and Technology Fund from the Hong Kong SAR Government (Project No. ITS/117/01).

9. References

1. Fung, T. Y. and Meng, H., “Concatenating Syllables for Response Generation in Domain-Specific Applications”, Proc. of ICASSP, 2000.
2. Meng, H. et. al., “CU VOCAL: Corpus-based Syllable Concatenation for Chinese Speech Synthesis across Domains and Dialects”, Proc. of ICSLP, 2002.
3. Fung, T. Y. and Meng, H., “The Effect of Tonal Context on Cantonese Concatenative Speech Synthesis”, Proc. of ISCSLP, 2002.
4. Meng. H. et al., “Recent Enhancements in CU VOCAL for Chinese TTS-Enabled Applications”, Proc. of EuroSpeech, 2003.
5. Meng. H. et. al., “CU VOCAL Web Service: A Text-to-speech Synthesis Web Service for Voice-enabled Web-mediated Applications”, Proc. of WWW, 2003.
6. Hoffmann, R. et al., “A Multilingual TTS System with less than 1 MByte Footprint for Embedded Applications”, Proc. of ICASSP, 2003.
7. Speech Synthesis Markup Language (SSML) 1.0, http://www.w3.org/TR/speech-synthesis/
8. Fung, T. Y. et al., “Prosody and Style Controls in CU VOCAL using SSML and SAPI XML tags”, Proc. of ISCSLP, 2004
9. Expat XML parser, http://expat.sourceforge.net/
10. Salton, G. and McGill, M., “Introduction to Modern Information Retrieval”, McGraw Hill, New York, NY, 1983.
11. Ogg Vorbis, http://www.vorbis.com/
12. MP3 Tech, http://www.mpe-tech.org/