EFFICIENT AND SCALABLE METHODS FOR TEXT SCRIPT GENERATION IN CORPUS-BASED TTS DESIGN

Chih-Chung Kuo and Jing-Yi Huang
Industrial Technology Research Institute (ITRI)
Hsinchu, Taiwan
cck@itri.org.tw

ABSTRACT
This paper proposes performance indices and search criteria for the text script generation in the design of corpus-based TTS systems. Based on the criteria a new search method is presented to solve the text selection problem more systematically and efficiently. Experiment results have shown that with the same hit rate of unit types the new method can reduce up to 40% of text script size in some cases. Furthermore, by control the weighting factor the covering rate of unit types can be increased to improve the robustness of the TTS system. Finally, the scalable and controllable design of the multi-stage search can produce various kinds of text scripts ideally suitable for the requirement of various kinds of corpus-based TTS systems.

1. INTRODUCTION
The synthesis units used in a text-to-speech (TTS) system have evolved from a small-sized inventory of units into a big corpus-based inventory [1]. The phonetic and prosodic features of a unit as well as of its context define a contextual unit. For any input text to the TTS system, the units with most similar phonetic and prosodic features to the target will be selected and concatenated to produce the synthetic speech. The stored speech segment of the units are expected to meet the target both in phonetic and in prosodic conditions so that the modification for synthesis can be as little as possible because any modification of the waveform will produce quality degradation.

Therefore, one of the key factors for a corpus-based TTS system is a well-designed text script for recording. The text script must contain sufficiently diverse types of unit so that in most cases a suitable unit can always be found to meet the requirements of the input target. On the other hand, however, the cost both for building a TTS system and for the storage requirement of the system depends on the size of the text script. Hence, the text script size cannot be expanded unlimitedly. It then comes up two problems. 1) How to define the efficiency of the text script? 2) How to generate an efficient text script? This paper investigates the problems and proposes some essential performance indices for the text script and methods to generate an efficient text script according to any given specifications.

Conventionally, there are two approaches to the text script generation. One is to emphasize the diversity of unit types in the inventory [2]. The other is to pursue the probability for the unit type of an input case to be found in the inventory [3]. The first approach tries to select the text containing richness of phonetic and prosodic features. The text script is usually selected from more than one corpus to search for various kinds of contextual combinations. Even sentences designed purposely by linguists are also used. The second approach represents the recent trend to use very large corpus. The occurring rates of the unit types in the large corpus are taken into account in text script generation so as to maximize the probability to hit the same unit type in synthesis. This paper clarifies the goals of the two approaches and proposes new integrated methods to produce better performance. Moreover, the goals can be clearly specified and exactly achieved if the corpus is affordable. This makes the generation of the text script controllable and scalable.

In Section 2, the problem will be defined formally and the performance indices and selection criteria for the problem will be presented. Based on the criteria, various search methods are given in Section 3. Experiments and results are shown in Section 4. Section 5 gives the conclusions.

2. PROBLEM & CRITERIA DEFINITION
2.1. Functions definition
Define the unit type function as follows:

\[ u = t(x) \]  
(1)

where \( u \) is the unit type to which the unit instance \( x \) belongs. Define two mapping functions of sets as follows:

The unit-type covering function:

\[ U = T(X) = \{ u = t(x) \mid \forall x \in X \} \]  
(2)

The unit-instance gathering function:

\[ X' = G(X,U) = \{ x' \mid \forall x' \in X \ and \ t(x') \in U \} \]  
(3)

where \( X \) is a set of unit instances and \( U \) is a set of unit types. Obviously, we have \( G(X,T(X)) = X \), or more generally, \( \forall X_i \subseteq X \Rightarrow G(X_i,T(X_i)) = X_i \subseteq X' \subseteq X \).

2.2. Sets definition
The problem can be clearly visualized in Fig. 1, where the sets are defined as follows:

\( X \) : the set of all unit instances in the corpus.
\( X_0 \) : the set of all unit instances in the selected text script.
\( U \) : the set of unit types covered by \( X \), i.e., \( U = T(X) \).
\( U_J \) : the set of unit types covered by \( X_J \), i.e., \( U_J = T(X_J) \).
\( X' \) : the set of all unit instances gathered by \( U_J \), i.e., \( X' = G(X,U_J) = G(X,T(X_J)) \).

It’s clear that \( X_i \subseteq X' \subseteq X \) and \( U_J \subseteq U \).
2.3. Problem description

The problem is to find the text script, $X_S$, to meet two virtually contradictory goals:

a) The text script should cover as many unit types as possible so that when any text is input to the TTS system there are suitable unit instances could be found for concatenation. However, the occurring frequency of each unit type differs dramatically, so the practical possibility for finding a match unit should also be considered.

b) The size of the text script (i.e. the amount of instances contained) should be as small as possible so that not only the processing cost of speech corpus could be less but also the memory requirement of the TTS system could be lower.

2.4. Performance indices

The first goal for the selected text script $X_S$ is to cover as many unit types as possible. Therefore, the first performance index can be the unit-type Covering Rate (CR) defined as follows:

$$r_C = \frac{|U_S|}{|X|} = \frac{|\{X,T(X_S)\}|}{|X|} \leq 1$$

The notation $|U_S|$ represents the size of the set $U$, i.e., the number of the elements in the set $U$.

As mentioned before, the occurring rate of each unit type is quite different. Thus, the total instances gathered by the $U_S$ must be considered, too. Thus, the second performance index, the unit-type Hit Rate (HR) is defined as follows:

$$r_H = \frac{|X_C|}{|X|} = \frac{|\{X,T(X_S)\}|}{|X|} \leq 1$$

2.5. Selection criteria

The first goal is therefore to maximize the CR or the HR. On the other hand, the second goal mentioned is to minimize the size of the text script, i.e., $|X_S|$. To combine the two contradictory goals together, we define the following criteria for the selection of the text script:

Covering-rate Efficiency:

$$\eta_C = \frac{r_C}{|X|} = \frac{|U_S|}{|X|}$$

Integrated Efficiency:

$$\eta_I = w \cdot \eta_H + (1-w) \cdot \eta_C = 1 \left( \frac{w}{|X|} \left[ |X| \cdot \mu \cdot |U_S| \right] \right)$$

where $\mu = \frac{|X|}{|X|} \geq 1$ is the average number of instances per unit type, and $w$ is the weighting factor with the value $0 \leq w \leq 1$.

3. SEARCH METHODS

Although the corpus is represented as a set of unit instances above, the practical corpus is made up of sentences of text. The minimal unit for recording is a sentence. This means that the text script is a list of sentences that were selected from the corpus one by one. Therefore the generation of the text script is actually a search problem that tries to select the best possible list of sentences from the corpus.

3.1. Best sentence-efficiency search

The logical search criterion is the efficiency index of Eq. (8). For each un-selected sentence in the corpus, the temporary “accumulated efficiency” can be computed with the formula in Eq. (8). However, the better guess to achieve the global optimum is to select the sentence with the best efficiency except for the unit types already being selected before this search. That is, if the $X_S$ is the set of unit instances of the sentence and the $U_S$ is the set of unit types contained in the sentence except for those already being covered, the formula in Eq. (8) could be used as the selection criterion.

3.2. Termination criteria

The selection loop can be terminated based on many criteria:

- $|X_S|$: instance size. The search can stop when the selected text script has achieved a predefined size. For core unit search, the $|X_S|$ could represent the number of selected instances per unit type. Some floor value of instance size for each unit type could be defined to assure a minimal number of instances being selected for each core unit.
- $r_H$: hit rate. This is useful because we can control the hit rate of the resulting TTS inventory.
- $r_C$: covering rate of unit types.
- $r_I = w \cdot r_H + (1-w) \cdot \mu \cdot r_C$ : integrated index of hit-rate and covering-rate.

The criteria above can be used in any combination according to practical consideration. Different criteria can also be used in different stages of multi-stage search described below.

3.3. Scalable multi-stage search

The definition of unit types can range dramatically from a few context-independent units to huge amount of contextual units.
Different requirements for each kind of unit type class must be considered. Therefore, a multi-stage search method is designed to generate a more balanced text script. Usually, the fewer core unit types require a better type covering and should be selected first. This is because the cost for a core unit missing is higher. For robust consideration, the core unit types should be covered as many as possible. On the other hand, the larger amount of variant unit types expect better hit rate to achieve higher average performance and usually be searched in a latter stage.

The whole search algorithm is very general and flexible. Many different unit type classes can be used in any stage. Therefore, the dimension and resolution of the unit class can be scalable. Many criteria can be used to control the generated text script to meet any pre-defined specification. This implies that the performance and cost can be scalable and precisely controllable.

## 4. EXPERIMENTS

### 4.1. Source corpus

The source corpus in our experiments contains two parts. A smaller part is a phonetically balanced corpus consisting of manually collected or designed sentences that cover all 413 Mandarin syllables. A much larger part of the corpus contains sentences extracted from various materials in real life, including articles, newspapers, textbooks, dialog, interview, etc. The size of the final corpus, |X|, is 6,621,809 syllable instances, which is distributed in 617,734 sentences.

### 4.2. Unit types

Mandarin Chinese TTS is the target system of this paper. The 413 Mandarin syllables are chosen as the basic synthesis unit because a Chinese character is a monosyllable. Starting from the basic unit, different degrees of expansion of the unit types can be defined based on various phonetic and prosodic features about the unit.

Table 1 shows the features used for defining unit types in our experiments. The pronunciation of each Chinese character is specified by both a syllable and a tone. The intra-word and intra-sentence features are mainly about the syllable position inside a word and the word position inside a sentence. The words could be lexical words or even better prosodic words [4].

<table>
<thead>
<tr>
<th>Self features</th>
<th>Context features</th>
<th>Phonetic</th>
<th>Prosodic</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Syllable</td>
<td>Tone</td>
<td>Must</td>
</tr>
<tr>
<td>Neighbor</td>
<td></td>
<td>IPhone</td>
<td>RPhone</td>
<td>R Tone</td>
</tr>
<tr>
<td>Right</td>
<td></td>
<td>LTone</td>
<td>R Tone</td>
<td></td>
</tr>
<tr>
<td>Intra-Word</td>
<td></td>
<td>IWord</td>
<td>Should</td>
<td></td>
</tr>
<tr>
<td>Intra-Sentence</td>
<td></td>
<td>ISent</td>
<td>May</td>
<td></td>
</tr>
</tbody>
</table>

*Table 1: Features for defining unit types*

Any a unit type can be specified by a feature vector consisting of various dimensions of features. The feature vector with the features of the unit itself is called **Unit Vector** (UV) in this paper. On the other hand, the **Context Vector** (CV) consists of context information of a unit. Therefore, context-dependent unit is specified by **Contextual Unit Vector** (CUV), which is concatenated by UV and CV.

The size of the feature vector space depends on the resolution of each feature dimension. Table 2 shows some typical values. Three typical unit classes, CU2, CU3, and CU4, are used in our experiments. The practical number of unit types contained in the corpus for these three unit classes are 912,415, 1,418,914, and 1,673,051, respectively.

<table>
<thead>
<tr>
<th>Unit class</th>
<th>UV</th>
<th>U1</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>CU2</th>
<th>CU3</th>
<th>CU4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syllable</td>
<td>413</td>
<td>413</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>413</td>
<td>413</td>
<td>413</td>
</tr>
<tr>
<td>Tone</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>LPhone</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>14</td>
<td>17</td>
<td>11</td>
<td>14</td>
<td>17</td>
</tr>
<tr>
<td>RPhone</td>
<td>1</td>
<td>1</td>
<td>22</td>
<td>26</td>
<td>29</td>
<td>38</td>
<td>26</td>
<td>29</td>
<td>38</td>
</tr>
<tr>
<td>LTone</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>RTone</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>IWord</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>9</td>
<td>4</td>
<td>4</td>
<td>9</td>
</tr>
<tr>
<td>ISent</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

*Table 2: Size of feature-vector space*

### 4.3. Experiment results

#### 4.3.1. 2-stage search with different unit classes

The simplest multi-stage search is to search for U1 unit in the first stage and the CU2–CU4 in the second stage. The U1 represents the core unit types, which are context-independent and are essential for the completeness of the synthesizer. The CU2–CU4 class expands the unit types into context-dependent units, which are expected to cover various phonetic and prosodic contexts so as to improve the synthetic speech quality.

In the first stage, the weight w is 0 for emphasizing the covering rate and the termination criterion is to select a minimal number of instances for each unit type. In the second stage, the weight w is 1 to pursue the maximal hit rate. The performance results are given in Fig. 2. The search method described in [3] is also implemented and tested for comparison. It's clear that our results (denoted as ITRI) outperform the prior art (denoted as MS) in hit rate and even in covering rate with the same text script size. The results also show that the hit rate and covering rate descend with the space size of the unit class.

#### 4.3.2. 2-stage search with different weighting factors

Fig. 3 gives the result of U1-CU2 2-stage search with the weighting factor w of 5 values in the CU2 stage. It’s clear that the covering rate can be increased when w approaching 0. The hit rate decreases only slightly except for w = 0.

#### 4.3.3. 3-stage search

A 3-stage search method is given in Table 3 as an example. Through this kind of design, we can obtain the text script that contains unit types of various degrees of significance with specified hit rate or covering rate.

### 5. CONCLUSIONS

In this paper, we have defined the performance indices and the search criteria for the text script generation, based on which
new search methods have been proposed to solve the problem more systematically and efficiently. The experiments have shown that the new search method can generate very efficient text scripts. With the same hit rate, the new method can reduce about 40% of text script size in some case (CU4-ITRI vs. CU4-MS at hit rate above 0.4 in Fig. 2). Furthermore, by control the weighting factor the covering rate can be increased to improve the robustness of the inventory. Finally, the scalable and controllable design of the multi-stage search can produce various kinds of text scripts ideally suitable for the requirement of various kinds of corpus-based TTS systems.

### Table 3: A 3-stage search example

<table>
<thead>
<tr>
<th>Stage</th>
<th>Unit</th>
<th>w</th>
<th>Instance size</th>
<th>Covering rate</th>
<th>Hit rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>U1</td>
<td>0</td>
<td>10 per type</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>2</td>
<td>CU2</td>
<td>0.25</td>
<td>Unlimited</td>
<td>&gt;10%</td>
<td>&gt;50%</td>
</tr>
<tr>
<td>3</td>
<td>CU3</td>
<td>1</td>
<td>&gt;150 K</td>
<td>Unlimited</td>
<td>Unlimited</td>
</tr>
</tbody>
</table>

![Figure 2](image)

**Figure 2**: 2-stage search result with different unit classes.

![Figure 3](image)

**Figure 3**: Search result with different weighting factors.

6. **ACKNOWLEDGEMENT**

This paper is a partial result of the Project B31BXT5300 conducted by ITRI under sponsorship of the Ministry of Economic Affairs, Taiwan, R.O.C.

7. **REFERENCES**


