Customizing Base Unit Set with Speech Database
in TTS Systems

Yining Chen, Yong Zhao, Min Chu

Microsoft Research Asia, Beijing, China
{ynchen,yzhao,minchu}@microsoft.com

Abstract

In unit selection based speech synthesizer, defining a good unit set is crucial to the speech quality. In this paper, a method of customizing the TTS base unit set with a specific speech corpus is proposed. Multi-phoneme units are boosted from the initial phoneme-sized unit. A new multi-phoneme unit is added to the inventory based upon its own frequency count and the affected frequency count of other units. As a result, a large base unit set, which contains many multi-phoneme units, is formed when the speech corpus is large. While, for a small speech corpus, only a few bi-phoneme or tri-phoneme are found. Such a scalable base unit set makes it possible to achieve better smoothness in concatenation while maintain the naturalness of prosody. Evaluation results show that, after replacing the phone-sized base unit set with the customized set, the search speed is improved by 5 times and 59% preference score is obtained.

1. Introduction

In a concatenative TTS system, speech output is synthesized by concatenating small pre-stored speech segments. In such a system, a set of base units (BUs), which are the smallest constituents in the speech corpus need to be defined for each language. Typically, there are many instances with phonetic and prosodic variations for each BU in the unit inventory and a unit selection algorithm is used to select the most suitable segment sequence by minimizing a cost function. Defining a suitable base unit set (BUS) is very important for such systems.

Two conflicting requirements have to be balanced in defining a BUS. On the one hand, smaller units are preferred because they cover plenty of prosodic variations of each BU for getting natural prosody. On the other hand, larger units are preferred because they can preserve natural intra-unit coarticulation in the synthesized utterances.

The strategy for defining the BUS differs among languages due to the different phonological characteristics of languages. For languages which have a relatively small syllable set, such as Chinese which contains less than 2000 syllables, syllables are often used as the BUs. However, syllables become impractical for languages that have too many syllables to enumerate, e.g., English has more than 20,000 possible syllables and it is difficult to generate a closed list of syllables. In such a language, smaller BU such as phoneme, diphone, halfphone or the mixture of them are often used. However, such small BUS has shortcomings:

- Deploying smaller units cause more difficulties in precise unit segmentation, this is crucial for good quality of synthesized speech. For example, in English, the word ‘yes’ consists of three phonemes, /y/, /eh/ and /s/ (SAPI phone set [1] is used in this paper), where the boundary between /eh/ and /s/ can be labeled at ease, yet it is difficult to separate /y/ from /eh/ due to the flat transition between their formant tracks. Moreover, experiments show that if the co-articulation between two phones is strong, it will be difficult to smoothly concatenate two segments selected from different context.

Various approaches have been investigated for achieving non-uniform unit in concatenation. Hunt and Black [2] performed dynamic programming based search in a phoneme-sized unit lattice. They used a target cost to assure the similarity between the selected segment and the target unit, and used a concatenation cost to take care of the smooth concatenation at boundaries. As a by-product of the unit selection, speech segments that contain more than one BU are often tended to be selected from the speech corpus. Their work has been viewed as a standard unit selection method [3] and has many extensions [4,5]. Under such a non-uniform unit framework, units generated by the unit selection algorithm are non-uniform while the BUS for constructing the search space is still uniform. For example, phoneme-sized BUs are often used in synthesizing English and syllable-sized BUs are normally used in generating Chinese speech. Therefore, the search speed issue is the same and the smooth concatenation issue is only partially solved.

In this paper, multi-phoneme sized BUs are investigated. We proposed a method to customize the BUS with the speech database used in the TTS system. When the frequency count (FC) of a multi-phoneme string in the database is higher than a threshold, and merging them as one unit will not hurt the FC of other units in BUS, it will be added into the BUS. Counts of BU pairs are used to boost the multi-phoneme sized BU as much as possible. The customized BUS (CBUS) will be always used together with the speech database from which it is derived. If a new speech database is used, a new CBUS will be generated accordingly. When voice fonts are built from a large speech corpus, a large CBUS is usually identified. However, when personalized voice fonts [6] or domain specific voice fonts [7] are to be built, it is often the case that only a small speech corpus is available. Then, only a few multi-phoneme BUs will be used. Such a scalable BUS makes it possible to achieve smooth in concatenation while maintain the naturalness of prosody. Since the method is language independent, it is easy to generate an optimal CBUS for any new languages.
Another advantage of CBUS is that all the non-uniform unit selection algorithms described above can still be performed.

The paper is organized as follows. The baseline TTS system – Mulan [5] is introduced in Section 2. In Section 3, the method of generating CBUS is described. Evaluation results are given in Section 4. Finally, conclusions are drawn in Section 5.

2. Mulan TTS System

Mulan is a bilingual TTS system [5], which can switch between Mandarin and English smoothly and maintains smooth sentence level intonation even for mixed-language texts. Its unit selection module, which performs a dynamic search in a base unit lattice as illustrated in Fig. 1, is shared across languages. In Fig. 1, \( m \) is the number of unit in a sentence and \( n_i \) represents the instance number of unit \( i \). The cost function used in search is defined by equation (1)

\[
C = W_c C_c + W_d C_d
\]

where \( C_c \) is the concatenative cost and \( C_d \) is the target cost. \( W_c \) and \( W_d \) are their weights.

The computing complexity of the standard dynamic searching algorithm is \( O(m \times n^2) \), where \( n \) is the average number of candidates per unit. Longer BU means smaller \( m \) and \( n \). Yet, on average \( m/n \) is a constant. Hence, if \( m \) decreases to its half, the search speed can be improved by 8 times.

In Mulan, \( C_c \) is defined as a binary function. If two consecutive segments in the unit lattice are found to be consecutive segments in the speech database as well, \( C_c \) between them is set to 0, otherwise \( C_c \) is set to 1. Then a fast search algorithm can be applied and the computing complexity decreases to \( O(m \times n) \). Therefore, in Mulan, halving the \( m \) means 4 times speedup of the unit selection search.

The unit selection module is both language independent and unit independent. In conventional Mulan, tonal syllable is used as BU for Chinese and phoneme as BU for English. And the informal quality evaluation shows that the voice quality of synthesized Chinese is slightly better than that of English. Such differences are believed mainly to be caused by the difference in BUS. It is able to extend the phoneme-sized BUS of English to cover more multi-phoneme units. The detail algorithm is described in the next section.

The Mulan framework serves as the test bed of the CBUS.

3. Base Unit Customization

There are two requirements in BU customization. They are: each BU in BUS should have enough instances to cover the prosodic and phonetic variations; CBUS should be a super set of the initial BU. An algorithm for identifying new BUs, which is similar to the method of finding new words in Chinese [8], is introduced in Section 3.1. Such a process is repeated over the speech corpus until no more qualified new BU can be found. Then, all new BUs identified in these cycles are merged with initial BUS to form the new CBUS. The overall framework for generating CBUS is given in Section 3.2.

3.1. Identify new BUs

The basic idea of identifying new BUs is that if the FC of a pair of BUs in the speech corpus is larger than a threshold, \( X \), and the formation of this new BU will not hurt the FC of the two individual BUs, they will be merged into one BU. To achieve this goal, three BU tables are maintained. They are the initial BUS, the candidate BUS and the new BUS. As shown in Table 1 and 2, the BUs and their FCs are listed in them.

**Table 1**: Table for initial BUS.

<table>
<thead>
<tr>
<th>Base Unit Name</th>
<th>Frequency</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>ax</td>
<td>17096</td>
<td></td>
</tr>
<tr>
<td>ih</td>
<td>13352</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>7912</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>7840</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**Table 2**: Table for candidate BUS.

<table>
<thead>
<tr>
<th>Base Unit Name</th>
<th>Frequency</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>(ax n)</td>
<td>2551</td>
<td></td>
</tr>
<tr>
<td>(aa v)</td>
<td>1196</td>
<td></td>
</tr>
<tr>
<td>(h um)</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

The flowchart for identifying new BUs is given in Fig. 2. As illustrated in Fig. 2, the speech corpus, which has been phonetically transcribed, is first segmented with the initial BUS, and the FCs for both initial BUS and the re-segmented corpus are counted. The BU pairs that have FCs higher than \( X \) will be output to the candidate BUS. Then, the speech corpus is re-segmented with the merged list of initial BUS and candidate BUS, and FCs for both lists are updated by counting in the re-segmented corpus. At last, new BU candidates are verified one by one from low FC end. If any FC of the two individual BUs (who
form the new BU) is lower than X, the new BU will be removed from the candidate BUS and the FCs of the two individual BUs will be adjusted accordingly. After the emendation, the remaining new BUs in the candidate list will be incorporated into the BUS.

![Figure 2](image)

**Figure 2**: The flowchart for identifying new BUs.

The corpus segmentation algorithm and BU verification algorithm are introduced in Section 3.1.1 and 3.1.2.

### 3.1.1. Unit string segmentation

The speech corpus is initially transcribed with phoneme strings. In order to calculate the FCs of multi-phoneme BUs, the phoneme transcriptions have to be segmented into BUs, just like the word segmentation in Chinese and Japanese [9]. Since there may be ambiguities in the phoneme string segmentation, (for example /a b c/ can be either segmented into /ab c/ or /a bc/), the widely adopted Forward Maximum Matching algorithm in word segmentation [9] is extended to BU segmentation.

### 3.1.2. New BU verification

The verification starts from the low FC end of both the initial list and candidate list. It is iteratively operated as follows:

1. Sort both the initial BU list and the candidate BU list by their FCs descendingly. Delete all candidate BUs whose FCs are lower than a present threshold X. If the candidate list is empty, go to step 5.
2. Set $N_i = N$, $N$ is the number of BUs in the initial list. $u_i$ is a BU in the initial list.
3. If $FC(u_i) \geq X$, go to step 4. Otherwise, search in the candidate list from the end for a new BU, $(u_i, u_j)$, by uniting $u_i$ and another initial BU, say $u_j$. If found, delete the BU $(u_i, u_j)$ from the candidate list and update the FC of $u_i$ and $u_j$ with the equation (2) and (3). If not found, go to step 4.
   
   \[
   FC(u_i) = FC(u_i) + FC(u_i, u_j) \quad (2)
   \]
   \[
   FC(u_j) = FC(u_j) + FC(u_i, u_j) \quad (3)
   \]
4. $i = i - 1$, If $i = 0$, go to step 5. Otherwise, go to step 3.
5. End

### 3.2. Framework of generating CBUS

Since the new BUs are identified by looking for high frequency BU pairs, multi-phoneme BUs can be identified by doing so iteratively. The framework for generating final CBUS is shown in Fig.3. The initial BUS is set to phonemes set at the beginning. In each cycle, the initial BUS and the speech database is input to the BU identifying module, which outputs a new BUS. If the new BUS is not empty, it is to be added to the initial BUS and the identifying loop is repeated. If the new BUS is empty, i.e. no more qualified BU can be identified, the process is ended.

![Figure 3](image)

**Figure 3**: The framework of customizing base unit.

The final CBUS is shown in Table 3. It can be seen that the BU list increased after each cycle.

**Table 3**: The table for CBUS

<table>
<thead>
<tr>
<th>Base Unit Name</th>
<th>Frequency Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>sh ax n z</td>
<td>110</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>m ax n</td>
<td>123</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>ax n</td>
<td>680</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>ax</td>
<td>3023</td>
</tr>
<tr>
<td>ih</td>
<td>2413</td>
</tr>
<tr>
<td>t</td>
<td>2402</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

For example, the sentence “I have not forgotten him” is initially presented as “ay / h ae v / n aa t / fa x r - g aa - t ax n / h ih m /”. Each phone is a BU. Here, slash marks the word boundary and hyphen marks the syllable boundary. After
segmented with the final CBUS, the sentence looks like “ay / ( h ae v ) / ( n aa t ) / f ax r - g aa t - ax n / ( h ih m ) /”, where the phoneme strings in the bracket are new BUs.

4. Evaluation Test

The new CBUS is directly implemented in the Mulan system and is compared with the original system that uses phoneme-sized unit set. The speech database used for building the voice font includes 6413 sentences (about 5 hours of speech). In the baseline system, 60 phonemes are utilized as base units. The final CBUS obtained with the proposed method contains 363 multi-phoneme units, i.e. it has 423 units in total.

743 English sentences are synthesized with both BUS. The sentence length varies from 7 to 14 words. There are all together 5603 words and 24804 phonemes in them.

![Figure 4: The time consumed in searching.](image)

The performance test is implemented on a computer which has Intel P4 3.0G HT CPU. The time of the unit selection module is measured for both the phoneme base unit set (PBUS) and CBUS is shown in Fig.4. The process speed with the CBUS is 5 times faster than that with the PBUS.

![Figure 5: The result of preference test.](image)

Among the 743 sentences, 532 of them got the exactly same speech waveforms when they are synthesized with the two BUSs. The other 211 sentences have at least one different segment. The two synthetic versions are then presented to 3 subjects to do the AB preference test. The result is shown in Fig.5. 59% utterances sound better when they are synthesized with the CBUS. The difference between the two sets is significant (P<0.00001).

5. Conclusion and discussion

Although phoneme sized base units are commonly adopted by most TTS systems, we found that multi-phoneme base units are helpful if they have enough instances in the speech database. Larger units can improve the search efficiency and intra-unit coarticulation performance at the expanse of coverage of prosodic variation. In this paper, this tradeoff is optimized by customize base unit set driven from speech database.

The basic idea is that if two consecutive base units appear in the speech corpus for enough times and the merging of them will not hurt the appearance frequency count of the two individual units, they will become one single base unit. Preference test shows that 59% of utterances generated with CBUS sound better than those synthesized with PBUS. Moreover, the searching speed of speech synthesis is improved by more than 5 times.

In our experiments, the number of units per sentence is 33 on average when PBUS is used, while it decreases to 15 when CBUS is used. The estimated speech increase is \((33/15)^2 = 4.8\) times. It is very close to the testing result \((7.0/1.3 = 5.3\) times). In a typical dynamic search, the computing complexity is \(O(n^2)\). Then, the processing speed will increase by more than 10 times.

Beside the performance and voice quality, the proposed language independent data driven approach can be easily extended to applications like personalized, domain specific, or multi-lingual TTS.

6. References