# Duration Modeling and Memory Optimization in a Mandarin TTS System

*Jilei Tian, Jani Nurminen and Imre Kiss*

Multimedia Technologies Laboratory
Nokia Research Center, Tampere, Finland
{jilei.tian, jani.k.nurminen, imre.kiss}@nokia.com

10.21437/Interspeech.2005-604

## Abstract

Current speech synthesis efforts, both in research and in applications, are dominated by methods based on concatenation of spoken units. New progress in the concatenative text-to-speech (TTS) technology can be made mainly from two directions, either by reducing the memory footprint to integrate the system into embedded system, or by improving the synthesized speech quality in terms of intelligibility and naturalness. In this paper, we are focusing on the memory footprint reduction in a Mandarin TTS system. We show that significant memory reductions can be achieved through duration modeling and memory optimization of the lexicon data. The results obtained in the experiments indicate that the memory requirements of the duration data and lexicon can be significantly reduced while keeping the speech quality unaffected. For practical embedded implementations, this is a significant step towards an efficient TTS engine implementation. The applicability of the approach is verified in the speech synthesis system.

## 1. Introduction

The memory and computational resources in TTS applications implemented on embedded portable devices are inherently limited. Though these resources are expected to increase in the future portable devices, the number of applications required to run simultaneously is also very likely to increase. It is therefore very important to minimize the memory and computational requirements of any application intended to be run on a portable embedded device. Thus, the TTS models should be kept as small as possible while keeping performance at a sufficiently high level. In our Mandarin TTS system, the memory usage of data can be divided into lexicon, prosody and voice. There has been a considerable amount of research effort directed at the problem of compressing the voice data inventory for TTS by using state-of-the-art speech coding techniques. In the paper, we are targeting on optimizing lexicon and prosodic data.

Like many other speech and language processing techniques, such as automatic speech recognition (ASR), TTS systems typically require a lexicon [1]. In the case of Mandarin Chinese, the lexicon data typically requires very high run-time memory footprint and time-consuming search. This is due to the fact that there are 20,901 Chinese characters (closed set) defined in the Unicode chart [2] and more than 100,000 Chinese words (open set). Also, many characters have multiple pronunciations and part of speech (POS). POS is a primitive form of linguistic theory that posits a restricted inventory of word-type categories such as noun, verb, etc. This paper proposes a novel approach that significantly reduces the run-time memory, the search times and the lexicon data size without loss of text processing accuracy.

In concatenative speech synthesis systems, unit selection plays a critical role in reaching high-quality synthetic speech [3][4]. In our implementation, the unit selection is based on the output of the prosodic model as can be seen from Figure 1. The template based prosodic model for syllables includes context information $\mathbf{c}_{ij}$, pitch contour $\mathbf{p}_{ij}$ and duration information $d_{ij}$ of $j$-th instances of $i$-th syllables. In the application, for a given text, the context features $\mathbf{c}_i$ of the $i$-th syllable are extracted from the text through text analysis. Using the distance between the context features taken from the text and the context features pre-trained and stored in the prosodic model, a target pitch contour and duration of $j^*$-th instance in $i$-th syllable are selected so that the following distance is minimized,

$$j^* = \arg\min_j \{d(\mathbf{c_i}, \mathbf{c_{ij}})\}. \quad (1)$$

Then, the selected pitch contour and duration information are used to select the best acoustic unit $k^*$-th instance of $i$-th syllable from database inventory,

$$k^* = \arg\min_k \{d(\{\mathbf{p_{ij^*}}, d_{ij^*}, ...\}, \{\mathbf{p}_{ik}, d_{ik}, ...\})\}. \quad (2)$$

In total, there are 1,678 syllables and 79,232 instances in the prosodic model of the Mandarin TTS system used in this paper. On average, there are 47 instances for each syllable. The duration data takes about 155KB when two bytes are assigned to each duration value.

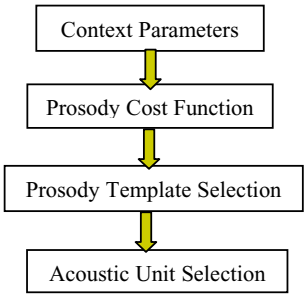


*Figure 1. Flowchart of acoustic unit selection.*

The concept of eigenpitch was proposed earlier to efficiently represent the pitch information [5]. Thus in this paper, we are only focusing on duration modeling to further reduce the memory footprint of the prosody model. The results in our Mandarin TTS system show that the duration data in the prosody model can be reduced from 150KB to 3KB, while keeping the error statistically on an acceptable level.

In this paper, the method to optimize the run-time memory of a Chinese lexicon is briefly described in Section 2. Then, Section 3 introduces the statistical duration model used for reducing the size of the prosody model. The practical results achieved in our Mandarin TTS system are presented in Section 4. Finally, conclusions are given in Section 5.

## 2. Lexicon

Before describing the method of memory optimized Chinese lexicon, some definitions and background information are presented. A Chinese character is a basic written unit and denoted by two bytes in Unicode. The whole set of Chinese characters contains 20,901 characters including the simplified and the traditional character sets. In the Unicode chart, Chinese characters are represented using the range from 4E00 to 9FA5. A Chinese character has ambiguous meanings and may have multiple pronunciations. The pronunciation of a Chinese character is presented by monosyllable pinyin. A Chinese word is a sequence of Chinese characters without separators. Suppose given a word consisting of $N$ characters, it can be denoted as an array: $word[0..N-1]$, where $word[0]$ stands for the Unicode value of the first character in the word. In this paper, an item is defined to represent all information related to a single word, including the character string, the pronunciation, POS, etc.

First of all, the Chinese lexicon data is arranged in the ascending order by Unicode values of the words. For example, the data can be stored in the stream of:

$Basic\_info \rightarrow item[0] \rightarrow \ldots\ldots \rightarrow item[Word\_Number-1]$,

where *Basic_info* contains information such as version, number of words (*Word_Number*), number of POS, etc. Each *item* is presented in the stream of:

$word \rightarrow pronunciation \rightarrow pos \rightarrow multiple\_pronunciations \rightarrow multiple\_pos$;

Ascending order means that for any $i$ between 0 and number of words, we have

$$item[i].word[0] \geq item[i-1].word[0],$$

where $item[i].word$ is Unicode sequence and $word[i]$ is Unicode value of the $i$+1-th character in the *word*.

When processing the lexicon data, instead of loading the whole lexicon data into run-time memory, a location array *location* is established. *Character_Number* denotes the total number of different characters (20,901 in Mandarin Chinese) and $location[i]$ indicates the length from the beginning of the lexicon data file to the first *item* whose first character has Unicode value derived from $i$. The location value can be extracted from lexicon data as shown in the pseudo-code.

```
For i = 0 to CHARACTER_NUMBER-1{
  location[i] = -1
}

unicode_value = item[0].word[0]
location[unicode_value - UNICODE_START] = location of item[0]

For i = 1 to CHARACTER_NUMBER-1{
  if item[i].word[0] and ≠ item[i-1].word[0]
  then{ unicode_value =item[i].word[0]
    location[unicode_value - UNICODE_START] = location of item[i] }
}
```

Searching any given word from the lexicon file is very fast. The first step is to locate the lexicon entries that contain a matching initial character (data from the corresponding location value to the next location value). The next step is to load in the data having this initial character and to search for the correct entry. The loaded data is usually very small and binary search can be applied to find matching item to given word within very small range. Alternatively, the search can be performed sequentially during loading. With this technique, it is possible to limit the size of the memory block needed for the loaded data to the maximum size of a single lexicon entry.

The location data can, of course, be extracted on-line during the initialization phase. However, the location data should preferably be extracted during the off-line processing of the lexicon data, and then stored as a part of the lexicon data. This process does not necessarily increase the overall lexicon data size as we will show in Section 4. If the location data is stored in the lexicon data, we already know the first character of every word, so the first character becomes redundant and can be omitted.

## 3. Statistical duration modeling

For the compression of the duration information in the prosody model, we propose a duration modeling approach. The technique is based on the fact that it is possible to sort the entries in the order of increasing duration value. Furthermore, we can extract statistical parameters that describe the behavior of the duration data. After this, all the duration values can be estimated by using the statistical parameters, index in the sorted array and the pre-assumed probability model. A very high compression rate can be reached by storing only the extracted statistical parameters, instead of the original duration values.

The duration information in our system is used for selecting the best acoustic unit. Due to the following issues, the accuracy required for the durations is not very critical.

1. In the prosodic model, the duration extracted from the training speech sample may not be very accurate because of the nature of speech (it is hard to find the ending of fricative, nasal, etc), and annotation is far from error-free.
2. In the unit selection, the duration distance is used along with many other distances. Compared with the pitch distance and the acoustic distance, the duration distance has a lower contribution to the unit selection.
3. It is not important to store perfectly accurate duration information in the prosody model since even the original duration values might not perfectly model the natural durations for the target speaker. (Due to the variations that might have happened during the recordings.)

As introduced above, the target duration is found by using the context distance in equation (1). Then the found target duration is used to find the best acoustic unit in equation (2). Since the search is always performed for a given syllable, memory can be saved if we only use statistical parameters extracted from all the instances of each syllable in the prosodic model; instead of using the instance based duration data, the syllable-based duration parameters are used. A probability model is applied to model the duration for each syllable.

In the original prosodic model, the entry of $i$-th syllable and $j$-th instance can be represented as

$\hat{\mathbf{e}}_{ij}=(\mathbf{c}_{ij}, \mathbf{p}_{ij}, d_{ij})$, (3)

Suppose we have $M$ instances for the syllable $i$ in the prosodic model. The mean and the standard deviation of durations for a given syllable are calculated as $m_d$ and $\sigma_d$, respectively. $P(d)$ stands for its probability distribution. Then all the entries within each syllable can be sorted based on the duration in increasing order. We use $\mathbf{e}_{ij}$ to represent sorted entries.

Now we propose the optimal method to estimate the duration $d_{ij}$ by using model parameters $m_d$ and $\sigma_d$. By using the proposed probability model, $d_{ij}$ can be completely removed since they can be estimate by $m_d$ and $\sigma_d$. For simplicity, assume we have $M$ duration values in the sorted order: $d_1<d_2<\ldots\ldots<d_M$, and estimated as $\hat{d}_j$. We have

$$m_d = \frac{1}{M}\cdot\sum_{j=1}^{M} d_j \quad \text{and} \quad \sigma_d = \sqrt{\frac{1}{M-1}\cdot\sum_{j=1}^{M}(d_j - m_d)^2} \tag{4}$$

Assume $L_j = \hat{d}_j - \hat{d}_{j-1}$, Moreover, let the lower and upper bounds of duration be $d_l$ and $d_h$. Then, the following condition should be approximately met

$$P(d_j)\cdot L_j = \text{Constant} \quad \Rightarrow \quad L_j = \frac{\text{Constant}}{P(d_j)} \tag{5}$$

Clearly

$$\sum_{j=1}^{M} L_j = d_h - d_l\,. \tag{6}$$

By inserting equation (5) into (6), we have

$$\text{Constant} = \frac{d_h - d_l}{\sum_{j=1}^{M}\frac{1}{P(d_j)}} \tag{7}$$

Thus, the duration values can be recursively estimated by

$$\hat{d}_{j,new} = \hat{d}_{j-1,new} + \frac{\frac{1}{P(d_{j-1,old})}}{\sum_{j=1}^{M}\frac{1}{P(d_{j-1,old})}}\cdot(d_h - d_l)\,. \tag{8}$$

Two probability models are analyzed and tested.

*Uniform probability model*

For the uniform probability model, Equation (8) can be re-written as

$$\hat{d}_j = \hat{d}_{j-1} + \frac{1}{N}\cdot(d_h - d_l) = d_l + \frac{(d_h - d_l)}{N}\cdot i \tag{9}$$

The estimated duration can be calculated efficiently without recursion.

*Gaussian probability model*

Equation (8) is re-written as

$$\hat{d}_{j,new} = \hat{d}_{j-1,new} + \frac{e^{\frac{1}{2}(\frac{d_{j,old}-m_d}{\sigma_d})^2}}{\sum_{j=1}^{M} e^{\frac{1}{2}(\frac{d_{j,old}-m_d}{\sigma_d})^2}}\cdot(d_h - d_l) \tag{10}$$

The initial duration values used in the probability model are equally spaced between the lower and the upper bound range. As seen in equation (10), it is computationally expensive.

It is worth to mention that it is also feasible to use curve fitting to the sorted duration curve shown in Figure 3. By duration curve fitting, some polynomial, spline, or even vector quantization can be applied. In theory, this approach can be equivalent to the probability model, but may offer a lower computational complexity and it might also be used for controlling the fitness of the data.

## 4. Experiments

### 4.1. Lexicon

To better understand the outcome of the proposed memory optimization method, an analysis was carried out on the Chinese lexicon used in our text-to-speech system. The lexicon data contains 20,901 Chinese characters (full Unicode set), 92,901 words and 68 POS. The size of the lexicon data in the file is 1,119,707 bytes. After loading the lexicon data into data structures in the run-time memory, the size is 4,771,860 bytes. Since the size of lexicon data is about 1MB, 3 bytes (0-4MB) are sufficient to represent the location information. The size of the location array is

20,901 X 3 bytes = 62,703 bytes = 61.2 KB;

The gain on the run-time memory usage between original and the proposed methods is 76 times. The search complexity is also reduced because each search is only conducted on a very small amount of data, rather than on the whole database as originally.

Regarding the lexicon data, the location information takes a 61.2 KB overhead, but all the first characters can be removed with size of about 180 KB leading to 120 KB saving. Thus, in addition to other advantages, we also obtain about 10% size reduction of the lexicon data file.

### 4.2. Duration

To demonstrate the properties of the proposed method, we carried out practical experiments using the prosodic model in our internal Mandarin TTS system, consisting of 79,232 instances and 1,678 syllables from a single female speaker. For each of the syllables, the durations are first automatically extracted and then manually validated. Finally all the entries within each syllable are sorted based on the duration values in increasing order. The mean and the standard deviation are calculated for each syllable. Three scenarios are tested.

1. Only the mean is used for each syllable, denoted as 'Baseline';
2. The mean and the standard deviation are used for each syllable, with the uniform probability duration model, denoted as 'Uniform';
3. The mean and the standard deviation are used for each syllable, with the Gaussian probability duration model, denoted as 'Gaussian';

Table 1 compares the performance among the three schemes. The Gaussian scheme performs best with the smallest average error and variance. It can get explained from Figure 2. The histograms of the durations for all syllables and a single syllable exhibit Gaussian-like distribution. Therefore the Gaussian probability model can fit the data better than the uniform probability model. Since only the mean is used for the baseline, it models the duration even worse due to the lack of statistical parameters. Figure 3 shows the duration values of the sorted instances of a given syllable (solid) and estimated values by uniform modeling (dotted), Gaussian

modeling (dashed). The estimated error of the duration values is reasonable small, and can be improved by better modeling.

| | Baseline | Uniform | Gaussian |
|---|---|---|---|
| Mean of absolute error | 26.28 | 7.97 | 6.59 |
| Standard deviation of absolute error | 12.78 | 5.22 | 4.36 |

*Table 1. Performance comparison of duration modeling among Baseline, Uniform and Gaussian models.*

Figure 3 shows an example of durations with the original values and the estimated values. The original duration values are taken from a single syllable selected arbitrarily from the prosodic model. Both uniform and Gaussian models are used to estimate the duration values. Here it is also possible to verify that Gaussian modeling gives better estimates of duration values than uniform modeling.

With the proposed approach, only the mean and the standard deviation need to be saved for each syllable. By assigning 1 byte for mean and 1 byte for standard deviation, only two bytes are needed for modeling the durations of one syllable. Since there are 1,678 syllables, thus the total memory needed for the duration information is:

1678X2 bytes=3356 bytes=3.3KB

Originally, the duration information needs

79,232 instances X 2 bytes = 155KB.

The memory of duration information is reduced from the original 155KB to 3.3KB, with a very small loss in accuracy as stated in Table 1. Furthermore, according to our experiments, output speech quality is practically unaffected.

## 5. Conclusions

The paper has analyzed the lexicon and duration modeling and their compact representation in a Mandarin concatenative TTS system. With the proposed lexicon optimization, the run-time memory of lexicon is reduced from originally 4 MB to about 60 KB. Meanwhile the search is made faster and the overall size of the lexicon data is reduced as well.

The proposed statistical duration modeling approach leads to memory reduction from the original 155 KB to 3.3KB while still keeping the duration error statistically at an acceptable level. The key techniques are (1) to sort all the durations for each syllable, then (2) to use a probability model for the prediction of durations. Though the Gaussian model was shown to provide better performance, the uniform model has a very light computational load with acceptable error. Thus, the uniform scheme can be considered as a suitable trade-off between memory saving, computational complexity and performance.

It is concluded that the presented scheme can significantly reduce the memory size of the lexicon and prosodic data while keeping the error statistically at an acceptable level. The presented method can be utilized in Mandarin TTS system. Experimental results verified the usefulness of the approach.

## 6. Acknowledgements

This work has partially been funded by the European Union under the integrated project TC-STAR - Technology and Corpora for Speech-to-Speech Translation - (IST-2002-FP6-506738, http://www.tc-star.org).

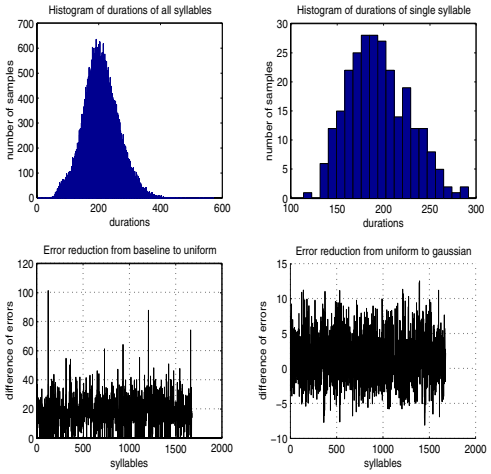


*Figure 2. The histograms of durations for the whole data set and for single syllable (upper subplots); the error differences between Baseline/Uniform and Uniform/Gaussian schemes (lower subplots).*

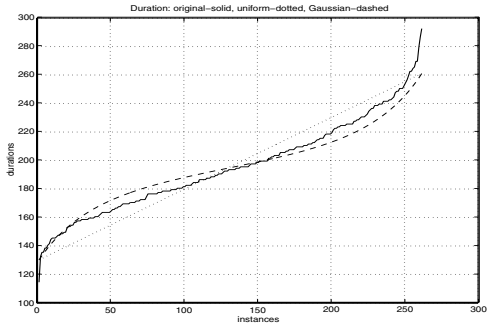


*Figure 3. The original duration values, extracted from one syllable, compared with the estimated values by uniform and Gaussian schemes.*

## 7. References

[1] Viikki, O., Kiss, I. and Tian, J., "Speaker- and Language-Independent Speech Recognition in Mobile Communication Systems", *International Conference on Acoustics, Speech, and Signal Processing*, Salt Lake City, USA, 2001.

[2] Unicode chart: http://www.unicode.org/charts/

[3] Dutoit, T., *An Introduction to Text-to-Speech Synthesis*, Kluwer Academic Publishers, Dordrecht, 1997.

[4] Huang, X., Acero, A. and Hon, H., *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*, Pearson Education, USA, 2001.

[5] Tian, J. and Nurminen, J., "On Analysis of Eigenpitch in Mandarin Chinese", The 4th *International Symposium on Chinese Spoken Language Processing*, Hong Kong, China, 2004.